MATHEMATICS OF COMPUTATION
VOLUME 39, NUMBER 160
OCTOBER 1982, PAGES 671–680

Computing in Permutation and Matrix Groups II: Backtrack Algorithm*

By Gregory Butler

Abstract. This is the second paper in a series which discusses computation in permutation and matrix groups of very large order. The essential aspects of a backtrack algorithm which searches these groups are presented. We then uniformly describe algorithms for computing centralizers, intersections, and set stabilizers, as well as an algorithm which determines whether two elements are conjugate.

1. Introduction. Only since the pioneering work of C. C. Sims [13], [14] in computing intersections, set stabilizers, and centralizers in permutation groups has the backtrack algorithm been applied to group-theoretic problems. The computation of the automorphism group of a graph [11], Hadamard matrix [9], code [10], or group [12] as a group of permutations uses the backtrack algorithm, as does the computation of normalizers in permutation groups [2]. Here we present in a uniform manner a backtrack search of a permutation or matrix group, thus giving a neat description of the algorithms of Sims and their generalizations to matrix groups. An algorithm for testing conjugacy of elements, which is derived from the centralizer algorithm, is also given.

The computation of Sylow subgroups will be described elsewhere [4]. Work is in progress on the computation of the conjugacy classes of elements using the algorithms for testing conjugacy and for computing centralizers.

We assume the reader is familiar with the first paper [3] in the series, whose notation we will follow. If $T = [y_1,\ldots,y_r]$ is a sequence of points, then $T \cup [y]$ will denote the sequence $[y_1,\ldots,y_r, y]$. Throughout, G will be a permutation or matrix group acting faithfully on a finite set X, and $B = [x_1, x_2,\ldots,x_k]$ will be a base for G relative to which a strong generating set of G is known. The key to the efficiency of the backtrack algorithms is the appropriate choice of a base. Efficient algorithms for changing from one base to another are known (Sims [14], Butler [1]), so we may assume that B is appropriate to the problem at hand.

We first describe in Section 2 a backtrack search for an element of G with a given property P. This is extended in Section 3, after presenting the necessary theory, to an algorithm which constructs a strong generating set (relative to B) of a subgroup of

Received May 12, 1980; revised March 24, 1982.
1980 *Mathematics Subject Classification.* Primary 20–04, 20G40, 20E25.
Key words and phrases. Backtrack algorithm, permutation group, matrix group.
* This work forms part of the author's Ph. D. thesis at the University of Sydney under Dr. J. J. Cannon. The work was partially supported by the Australian Research Grants Committee.

0025-5718/82/0000-0373/$03.00

G. The implementation is discussed in Section 4, and the details for each specific type of subgroup are presented in Section 5. The conclusion in Section 6 is followed by tabulated performance results.

Tables I–V refer to results on a CDC Cyber 72 using our implementations which form part of the group theory system CAYLEY [5]. Any symbols in the column headings of the tables are explained in the relevant part of Section 5. For an explanation of the names of the groups see [8]. The group $?G_2(4)$ is a subgroup of $Sp(6,4)$, which we believe to be $G_2(4)$. The results in Table V are averages for testing conjugacy of permutations with the same cycle structure, in a situation where no change of base was necessary.

2. Backtrack Search. In this section we consider the problem of finding an element g of G with a given property P. We begin with some definitions. A sequence $T = [y_1,\ldots,y_r]$, $0 \leqslant r \leqslant k$, of distinct points is called a *partial (base) image*. If $r = k$, then T is *complete*. For any partial image T and subgroup H of G, define

$$H(T) = \{g \in H \mid [x_1,\ldots,x_r]^g = T\}$$

and, if T is incomplete, define

$$X_H(T) = \{y \in X \mid H(T \cup [y]) \neq \varnothing\}.$$

Similarly we define $P(T) = \{g \in G(T) \mid g \text{ has property } P\}$, and $X_P(T) = \{y \in X \mid P(T \cup [y]) \neq \varnothing\}$.

The backtrack algorithm runs through the partial images T using any knowledge of the set $X_P(T)$ to prune the search tree. Our knowledge of $X_P(T)$ comes from the following result, and from relating the choice of base to the property P.

PROPOSITION 1. *Let $T = [y_1,\ldots,y_r]$ be an incomplete partial image, and let $g \in G(T)$. Then*

(i) $X_G(T) = (x_{r+1}^{G^{(r+1)}})^g$, *and*

(ii) $X_P(T) \subseteq X_G(T)$.

The proof is easy and will be omitted.

The aim in developing an efficient backtrack algorithm is to be able to quickly determine a 'good' approximation, call it $\overline{X}_P(T)$, from T and P with $X_P(T) \subseteq \overline{X}_P(T)$. We can always assume that $\overline{X}_P(T) \subseteq X_G(T)$. Another feature which prominently affects the efficiency of the backtrack algorithm is the 'first point in the orbit' condition, as explained by the following result.

PROPOSITION 2. *Suppose K is a subgroup of G and that, for each g in G, either all or no elements of gK have property P. Let $T = [y_1,\ldots,y_r]$ be an incomplete partial image, and let $y \in X$. Then the set $P(T \cup [y]) = \varnothing$ if and only if, for all $h \in K_{y_1,\ldots,y_r}$, the set $P(T \cup [y^h]) = \varnothing$.*

Proof. It follows from the hypothesis that $g \in P(T \cup [y])$ if and only if $gh \in P(T \cup [y^h])$. □

One can always take K to be the trivial subgroup, but in all cases of interest—especially in Section 3—a nontrivial subgroup K is known. We totally order X so that $x_1,\ldots,x_k$ are the first k points. There is an induced (lexicographical) order on each of G, B^G, and the set of partial images. Proposition 2 simply says that only the

first point of each $K_{y_1,\ldots,y_r}$-orbit has to be considered when extending T. In the cases discussed here, this restriction is used when $K = K_{y_1,\ldots,y_r}$, otherwise the orbits of a quickly-computed approximation to $K_{y_1,\ldots,y_r}$ are used, in order to avoid a time-consuming change of base for K from B to $[y_1,\ldots,y_r,\ldots]$.

We present the general algorithm for finding an element g with property P and then discuss the case of testing conjugacy of elements.

ALGORITHM BKTKS

BKTKS 1: [initialize with null sequence]
$r \leftarrow 0, T \leftarrow [\,]$.

BKTKS 2: [next level down]
$r \leftarrow r + 1$.
if $r > k$ then go to (5).
$Y_r \leftarrow \overline{X}_P(T)$.

BKTKS 3: [next partial image]
if $Y_r = \varnothing$ then go to (4).
$y_r \leftarrow$ first point of Y_r, $T \leftarrow [y_1,\ldots,y_r]$.
go to (2).

BKTKS 4: [backtrack]
$r \leftarrow r - 1$.
if $r = 0$ then stop (no element with property P exists).
$Y_r \leftarrow Y_r \setminus y_r^{K_{y_1,\ldots,y_{r-1}}}, T \leftarrow [y_1,\ldots,y_{r-1}]$
go to (3).

BKTKS 5: [image is complete]
let g be the unique element of $G(T)$.
if g has property P then stop else go to (4).

In testing conjugacy of g_1 and g_2 in G we search for an element with the property

$$P\text{: ``}g \text{ conjugates } g_1 \text{ to } g_2\text{''}.$$

In this case K is any subgroup of $C_G(g_2)$, for example $\langle g_2 \rangle$. The base is chosen to be compatible (in the sense of [13]) with the action of g_1 so that $x_r = x_{r-1}^{g_1}$ as often as possible. Then, for an incomplete partial image $T = [y_1, y_2,\ldots,y_{r-1}]$, we can take

$$\overline{X}_P(T) = \begin{cases} X_G(T) \cap \{y_{r-1}^{g_2}\}, & \text{if } x_r = x_{r-1}^{g_1}, \text{ and} \\ \{y \in X_G(T) \mid |y^{\langle g_2 \rangle}| = |x_r^{\langle g_1 \rangle}|\}, & \text{otherwise,} \end{cases}$$

since a conjugating element maps cycles of g_1 to cycles of g_2.

This approach restricts the number of complete images considered, particularly for elements with large cycles. Our experience while using the algorithm as part of a method to determine the conjugacy classes of a permutation group (cf. Table V) indicates that using $K = C_G(g_2)$ instead of $K = \{\text{identity}\}$ often leads to a 30–50% reduction in time. We found the algorithm to be very efficient.

3. Subgroup Construction. In the case where the elements with property P form a subgroup H, the previous algorithm can be modified to find generators of $H^{(k)}$, then generators of $H^{(k-1)}$, and so on. In this way a strong generating set of H relative to B is computed. Suppose $H^{(s+1)} \leq K \leq H^{(s)}$. Then, in searching for generators of $H^{(s)}$, we clearly have to consider only one element from each right and left coset of

K. The following result therefore justifies discarding points in steps 2 and 4 of algorithm BKTK described below.

PROPOSITION 1. (i) (Sims [13]). *An element g is the first element of Kg if and only if, for each i, $1 \leqslant i \leqslant k$, $x_i^g \leqslant y^g$ for every y in the $K_{x_1,\ldots,x_{i-1}}$-orbit of x_i.*

(ii) *Let $y_i = x_i^g$ for $i = 1,2,\ldots,k$. Then g is the first element of gK if and only if, for each i, $1 \leqslant i \leqslant k$, y_i is first in the $K_{y_1,\ldots,y_{i-1}}$-orbit of y_i.*

Proof. The results follow from the one-to-one correspondence of Kg with B^{Kg} and of gK with $(B^g)^K$, and the fact that $x_1,\ldots,x_k$ are the first points of X.

(i) Suppose there exists an integer $i \leqslant k$ and a point $y \in K_{x_1,\ldots,x_{i-1}}$-orbit of x_i such that $y^g < x_i^g$. Let $h \in K_{x_1,\ldots,x_{i-1}}$ mapping x_i to y. Then $x_j^{hg} = x_j^g$, for $j < i$, and $x_i^{hg} = y^g < x_i^g$. Hence $hg < g$, and g is not first in Kg. Conversely, suppose that g is not first in Kg. Let $h \in K$ such that $hg < g$. If $x_i^{hg} = x_i^g$ for $i = 1,\ldots,k$, then $hg = g$. Hence, let i be the first integer for which $x_i^{hg} < x_i^g$. Then $h \in K_{x_1,\ldots,x_{i-1}}$ and, if $y = x_i^h$, then $y^g < x_i^g$.

(ii) Suppose there exists $i \leqslant k$ and $y \in K_{y_1,\ldots,y_{i-1}}$-orbit of y_i such that $y < y_i$. Let $h \in K_{y_1,\ldots,y_{i-1}}$ such that $y_i^h = y$. Then $x_j^{gh} = y_j^h = y_j$, for $j < i$, and $x_i^{gh} = y_i^h = y < y_i = x_i^g$. Hence $gh < g$, and g is not first in gK. Conversely, suppose that $h \in K$ and $gh < g$. Let i be the first integer such that $x_i^{gh} < x_i^g$. Then $h \in K_{y_1,\ldots,y_{i-1}}$ and $y_i^h < y_i$. □

A corollary of Proposition 1 (ii) is that once a generator g extending K has been found then no other generator of $H^{(s)}$ mapping x_s to x_s^g is required. Hence the value of r is set to $s + 1$ in step BKTK 5 after a generator has been found.

In algorithm BKTK the sequence $T = [y_1,\ldots,y_r]$; the integer s is the largest integer for which $y_i = x_i$ for all $i < s$; K is the subgroup of H already computed, $H^{(s+1)} \leqslant K \leqslant H^{(s)}$; and S is a strong generating set of K relative to B. On termination $K = H$. If L is a known subgroup of H, then L is used in step BKTK 6 to improve the computation.

ALGORITHM BKTK

BKTK 1: [initialize]
$K \leftarrow \{\text{identity}\}$, $S \leftarrow \varnothing$, $T \leftarrow B$.
for $r = 1,2,\ldots,k$, $Y_r \leftarrow \overline{X}_P([x_1,\ldots,x_{r-1}])$.
$s \leftarrow k$, $r \leftarrow s + 1$.

BKTK 2: [T and its descendants have been considered, so backtrack]
$r \leftarrow r - 1$.
if $r < s$ then go to (6).
$Y_r \leftarrow Y_r \setminus y_r^{K_{y_1,\ldots,y_{r-1}}}$, $T \leftarrow [y_1,\ldots,y_{r-1}]$.

BKTK 3: [consider next y_r]
if $Y_r = \varnothing$ then go to (2).
$y_r \leftarrow$ first point of Y_r, $T \leftarrow [y_1,\ldots,y_r]$.

BKTK 4: [consider descendants of T]
$r \leftarrow r + 1$.
if $r > k$ then go to (5).
$Y_r \leftarrow \overline{X}_P(T) \setminus \{y \in X \mid$ there exists $j < r$ such that $x_r \in x_j^{K^{(j)}}$ and $y < y_j\}$.
go to (3).

BKTK 5: [image is complete]
let g be the unique element of $G(T)$.
if g does not have property P then go to (2).
$S \leftarrow S \cup \{g\}$, $K \leftarrow \langle S \rangle$, $r \leftarrow s + 1$.
go to (2).

BKTK 6: [$K = H^{(s)}$. Construct $H^{(s-1)}$]
$s \leftarrow s - 1$.
if $s = 0$ then stop.
$S \leftarrow S \cup \{$generators of L which fix $x_1, \ldots, x_{s-1}\}$.
$K \leftarrow \langle S \rangle$.
$r \leftarrow s + 1$.
go to (2).

4. Implementation. For a permutation group G the implementation is straightforward. Subsets of X are represented as bit strings and an element g of $G(T)$ is stored to facilitate the computation of $X_G(T)$. The storage requirements over and above those necessary to frame the problem are k bit strings for the Y_r, an additional bit string for the orbit in step BKTK 2, one element, and the storage requirements of the group K.

For a matrix group G a different approach is used. The sequence T and the subsets Y_r of X are implicit. The information they contain is interpreted in terms of coset representatives in the basic transversals of G and explicitly stored as such. The sequence T of points is replaced by a sequence $I = [i_1, \ldots, i_r]$ of integers, where $u_{r,i_r} u_{r-1,i_{r-1}} \cdots u_{1,i_1} \in G(T)$ and u_{j,i_j} is the i_jth element of the jth basic transversal U_j. The set Y_r is replaced by a subset W_r of $\{1, 2, \ldots, |U_r|\}$, where the image of x_r under $u_{r,i_r} \cdots u_{1,i_1}$ runs over Y_r as i_r runs over W_r. The set W_r is stored as a bit string relative to $\{1, 2, \ldots, |U_r|\}$. Each of the products $u_{j,i_j} u_{j-1,i_{j-1}} \cdots u_{1,i_1}$, for $j = 1, \ldots, r$, is explicitly stored as an element in order to facilitate the computation of an element $g \in G(T)$ as T varies. The storage requirements are $2k$ bit strings (one for W_r and one for workspace at level r), k elements, the storage requirements of K, and a hash list of vectors for the orbit in step BKTK 2.

The specific algorithms which are based on the backtrack algorithm are implemented by supplying three additional routines for use by the general routine.

(1) A routine to select an appropriate base and to set up any information needed in the computation of $\overline{X}_P(T)$.

(2) A routine to compute $\overline{X}_P(T)$.

(3) A routine to test whether an element of G has property P.

An appropriate base as presented in the theoretical exposition of a specific algorithm may be quite long. However, it is always possible to implement the algorithm so that redundant levels of the base are not explicitly stored, thus making the actual length of the base manageable. Moreover, for permutation groups it is always possible to conjugate the setting of the problem in the symmetric group so that the appropriate base is $[1, 2, \ldots, k]$. Hence the natural order on X can be used.

The algorithms have been implemented using the dynamic storage manager STACKHANDLER [6] and they form part of CAYLEY [5].

5. Specific Cases of the Backtrack Algorithm. It is now a simple matter to describe a given backtrack algorithm by specifying (a) the property P, (b) the choice of base B, (c) the choice of L in step BKTK 6 and the approximation to $K_{y_1,\dots,y_{r-1}}$ in step BKTK 2 when $r > s$ (note that $r = s$ implies $K = K_{y_1,\dots,y_{r-1}}$), (d) $\overline{X}_P(T)$, and (e) any features of the implementation.

(5.1) Intersection.

(a) P: "g is in G and M".

(b) G and M have the same base B.

(c) $L = \{\text{identity}\}$ and $\langle s \in S \mid s \text{ fixes } y_1,\dots,y_{r-1}\rangle$ is the approximation to $K_{y_1,\dots,y_{r-1}}$.

(d) $\overline{X}_P(T) = X_G(T) \cap X_M(T)$.

Our experience indicates that the algorithm is very efficient on average. However, Hoffman [7] shows it has worst case behavior which is exponential in the degree.

(5.2) Set stabilizer.

(a) P: "g stabilizes $\{z_1,\dots,z_m\}$".

(b) $B = [z_1, z_2,\dots,z_m,\dots]$.

(c) $L = G_{z_1,z_2,\dots,z_m}$ and $\langle s \in S \mid s \text{ fixes } y_1,\dots y_{r-1}\rangle$ is the approximation to $K_{y_1,\dots,y_{r-1}}$.

(d) $\overline{X}_P(T) = X_G(T) \cap \{z_1,\dots,z_m\}$.

(e) As $G_{z_1,\dots,z_m} \leqslant H$ the backtrack algorithm searches the images of $[z_1,\dots,z_m]$ rather than the images of B. That is, $k = m$ and not the length of the base.

Our experience indicates that the algorithm is slow for large sets with small stabilizer. In general its application is restricted to the case where $m \leqslant 6$. The algorithm has not been implemented for matrix groups.

(5.3) Centralizer of an element.

(a) P: "g conjugates f to f".

(b) B is compatible with the cycles of f.

(c) $L = \{\text{identity}\}$ and the trivial group is the approximation to $K_{y_1,\dots,y_{r-1}}$.

(d)

$$\overline{X}_P(T) = \begin{cases} X_G(T) \cap \{y_{r-1}^f\}, & \text{if } x_r = x_{r-1}^f, \text{ and} \\ \{y \in X_G(T) \mid |y^{\langle f\rangle}| = |x_r^{\langle f\rangle}|\}, & \text{otherwise.} \end{cases}$$

(e) For permutation groups the set $\{y \in X \mid |y^{\langle f\rangle}| = |x_r^{\langle f\rangle}|\}$, for each suitable value of r, is stored as a bit string throughout the computation. For matrix groups the size of these sets is prohibitive and they are not stored. If $I = [i_1,\dots,i_{r-1}]$, $g = u_{r-1,i_{r-1}} u_{r-2,i_{r-2}} \cdots u_{1,i_1}$, and $x_r^{G^{(r)}} = \{v_1, v_2,\dots\}$, then

$$W_r = \begin{cases} \{i \mid v_i = x_{r-1}^{gfg^{-1}}\}, & \text{if } x_r = x_{r-1}^f, \text{ and} \\ \{i \mid v_i \text{ is in a cycle of } gfg^{-1} \text{ of length } |x_r^{\langle f\rangle}|\}, & \text{otherwise.} \end{cases}$$

There are further differences. The longest cycles of f are chosen to form the base if G is a permutation group. If G is a matrix group then the choice of cycles for the base is made from those cycles which contain a point of the existing base, and preference is given to cycles of subspaces in an attempt to minimize $|U_r|$. Our experience indicates that the algorithm is very efficient.

(5.4) Centralizer of a subgroup.

(a) P: "g centralizes F".

(b) B is compatible with the orbits of F. That is, $\{x_1, x_2, \ldots, x_i\}$, $\{x_{i+1}, x_{i+2}, \ldots\}, \ldots$ are orbits of F.

(c) as in (5.3).

(d)

$$\overline{X}_P(T) = \begin{cases} X_G(T) \cap \{x_{r-1}^{f_r}\}, & \text{if some } f_r \in F \text{ maps } x_{r-1} \text{ to } x_r, \\ \{y \in X_G(T) \mid |y^F| = |x_r^F| \text{ and } |y^{\langle f \rangle}| = |x_r^{\langle f \rangle}| \\ \qquad\qquad \text{for each generator } f \text{ of } F\}, & \text{otherwise.} \end{cases}$$

(e) The elements f_r are determined at the outset (from the Schreier vector of the F-orbit) and stored as words in the generators of F. For permutation groups the set $\{y \in X \mid |y^F| = |x_r^F| \text{ and } |y^{\langle f \rangle}| = |x_r^{\langle f \rangle}| \text{ for each generator } f \text{ of } F\}$, for each suitable value of r, is stored as a bit string throughout the computation. For matrix groups the first condition $|y^F| = |x_r^F|$ is not used (as orbits are relatively expensive to compute) and the sets are determined as in (5.3). The actual choice of the base is analogous to (5.3). Our experience indicates that the algorithm is very efficient.

6. Conclusion. The performance of the implementations of the specific algorithms is given in the appendix. There are two factors which particularly affect the efficiency. One is how closely $\overline{X}_P(T)$ approximates $X_P(T)$, and the other is the length of the K-orbits. Because of the latter, computing a group H of large order tends to be faster than computing a group of small order.

Although we have assumed that the matrices are over finite fields, there is in principle no obstacle to considering finite groups of matrices over other rings.

APPENDIX: Tables of Performance. All times are in CDC Cyber 72 seconds. All runs used the implementations in CAYLEY

TABLE I

Set stabilizer in permutation groups

G	$\lvert G\rvert$	$\lvert X\rvert$	m	$\lvert H\rvert$	TOTAL TIME	TIME TO CHANGE BASE
$L_5(2)$	$2^{10} \cdot 3^2 \cdot 5 \cdot 7 \cdot 31$	31	3	$2^8 \cdot 3^2$	0.98	0.18
M_{24}	$2^{10} \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 23$	24	3	$2^7 \cdot 3^3 \cdot 5 \cdot 7$	1.01	0.17
			6	$2^4 \cdot 3^3 \cdot 5$	1.24	0.17
			12	$2^4 \cdot 3 \cdot 5$	157.76	0.19
$L_3(13)$	$2^5 \cdot 3^2 \cdot 7 \cdot 13^3 \cdot 61$	183	2	$2^5 \cdot 3 \cdot 13^2$	3.65	0.78
			3	$2^5 \cdot 3^2$	3.56	1.00
			4	$2^3 \cdot 3$	5.69	2.78
			5	2	15.98	3.51
			20		> 500	

TABLE II

Centralizer of an element in permutation groups

G	$\|X\|$	$\|G\|$	$\|f\|$	$\|C_G(f)\|$	TOTAL TIME	TIME TO CHANGE BASE
$L(3,2)$	7	$2^3 \cdot 3 \cdot 7$	2	2^3	0.25	0.03
			3	3	0.21	0.03
$L(3,3)$	13	$2^4 \cdot 3^3 \cdot 13$	2	$2^4 \cdot 3$	0.48	0.10
			13	13	0.36	0.10
$L(3,4)$	21	$2^6 \cdot 3^2 \cdot 5 \cdot 7$	2	2^6	0.80	0.11
			7	7	0.52	0.12
$L(3,5)$	31	$2^5 \cdot 3 \cdot 5^3 \cdot 31$	2	$2^5 \cdot 3 \cdot 5$	0.10	0.22
			24	$2^3 \cdot 3$	0.74	0.25
$L(3,7)$	57	$2^5 \cdot 3^2 \cdot 7^3 \cdot 19$	2	$2^5 \cdot 3 \cdot 7$	1.65	0.38
			19	19	1.38	0.32
$L(3,8)$	73	$2^9 \cdot 3^2 \cdot 7^2 \cdot 73$	2	$2^9 \cdot 7$	2.25	0.45
			3	$3^2 \cdot 7$	2.53	0.53
$L(3,9)$	91	$2^7 \cdot 3^6 \cdot 5 \cdot 7 \cdot 13$	2	$2^7 \cdot 3^2 \cdot 5$	3.15	0.72
			40	$2^4 \cdot 5$	2.23	0.63
$L(3,11)$	133	$2^4 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11^3 \cdot 19$	2	$2^4 \cdot 3 \cdot 5^2 \cdot 11$	4.79	0.93
			120	$2^3 \cdot 3 \cdot 5$	2.61	0.78
$L(3,13)$	183	$2^5 \cdot 3^2 \cdot 7 \cdot 13^3 \cdot 61$	2	$2^5 \cdot 3 \cdot 7 \cdot 13$	6.13	1.20
			61	61	4.14	1.17
$H - S$	100	$2^9 \cdot 3^2 \cdot 5^3 \cdot 7 \cdot 11$	7	7	6.73	2.48
			2	$2^9 \cdot 3 \cdot 5$	5.63	1.38
$G(2,4)$	416	$2^{12} \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 13$	3	$2^6 \cdot 3^3 \cdot 5 \cdot 7$	14.44	2.62
			6	$2^2 \cdot 3$	30.67	4.53
			2	$2^8 \cdot 3 \cdot 5$	21.43	2.71
${}^2F_4(2)$	1755	$2^{10} \cdot 3^3 \cdot 5^2 \cdot 13$	2	$2^{10} \cdot 3$	100.63	15.18
			8	2^4	262.33	3.14

TABLE III

Centralizer of an element in matrix groups

G	dim V	|field|	$\|G\|$	$\|f\|$	$\|C_G(f)\|$	TOTAL TIME	TIME TO CHANGE BASE
$SL(3,2)$	3	2	$2^3 \cdot 3 \cdot 7$	2	2^3	0.76	0.38
				3	3	0.39	0.08
$SL(3,3)$	3	3	$2^4 \cdot 3^3 \cdot 13$	2	$2^4 \cdot 3$	1.73	0.96
				13	13	1.00	0.63
$SL(3,4)$	3	4	$2^6 \cdot 3^3 \cdot 5 \cdot 7$	2	$2^6 \cdot 3$	3.66	1.78
				21	$3 \cdot 7$	1.50	0.36
$SL(3,5)$	3	5	$2^5 \cdot 3 \cdot 5^3 \cdot 31$	2	$2^5 \cdot 3 \cdot 5$	3.90	2.25
				24	$2^3 \cdot 3$	2.28	1.45
$SL(3,7)$	3	7	$2^5 \cdot 3^3 \cdot 7^3 \cdot 19$	2	$2^5 \cdot 3^2 \cdot 7$	6.23	3.86
				19	$3 \cdot 19$	4.08	2.30
$SL(3,8)$	3	8	$2^9 \cdot 3^2 \cdot 7^2 \cdot 73$	2	$2^9 \cdot 7$	11.12	7.65
				3	$3^2 \cdot 7$	6.06	4.40
$SL(3,9)$	3	9	$2^7 \cdot 3^6 \cdot 5 \cdot 7 \cdot 13$	2	$2^7 \cdot 3^2 \cdot 5$	12.05	7.53
				40	$2^4 \cdot 5$	5.79	3.95
$SL(3,11)$	3	11	$2^4 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11^3 \cdot 19$	2	$2^4 \cdot 3 \cdot 5^2 \cdot 11$	14.67	8.81
				120	$2^3 \cdot 3 \cdot 5$	6.83	4.89
$SL(3,13)$	3	13	$2^5 \cdot 3^3 \cdot 7 \cdot 13^3 \cdot 61$	2	$2^5 \cdot 3^2 \cdot 7 \cdot 13$	23.22	14.00
				183	$3 \cdot 61$	12.76	8.41
$SL(4,4)$	4	4	$2^{12} \cdot 3^4 \cdot 5^2 \cdot 7 \cdot 71$	2	$2^{12} \cdot 3^2 \cdot 5$	25.50	17.37
				4	2^6	20.75	14.92
$SL(4,5)$	4	5	$2^9 \cdot 3^2 \cdot 5^6 \cdot 13 \cdot 31$	2	$2^8 \cdot 3^2 \cdot 5^2$	169.42	31.31
				124	$2^2 \cdot 31$	14.11	8.03
$SL(5,3)$	5	3	$2^9 \cdot 3^{10} \cdot 5 \cdot 11^2 \cdot 13$	2	$2^8 \cdot 3^4 \cdot 13$	77.64	57.77
				39	$2 \cdot 3 \cdot 13$	9.68	6.69
$Sp(6,3)$	6	3	$2^{10} \cdot 3^9 \cdot 5 \cdot 17 \cdot 13$	12	$2^2 \cdot 3^2$	28.51	10.18
				3	$2^5 \cdot 3^8$	48.27	16.72
$?G_2(4)$	6	4	$2^{12} \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 13$	6	$2^2 \cdot 3$	58.95	4.89
				2	$2^8 \cdot 3 \cdot 5$	85.17	6.94
$Sp(6,4)$	6	4	$2^{18} \cdot 3^4 \cdot 5^3 \cdot 7 \cdot 13 \cdot 17$	6	$2^4 \cdot 3$	84.06	31.77
				2	$2^{14} \cdot 3 \cdot 5$	233.38	17.67

TABLE IV

Centralizer of subgroup in matrix groups

G	$\|G\|$	$\dim V$	$\|\text{field}\|$	F	$\|F\|$	$\|C_G(F)\|$	TOTAL TIME	TIME TO CHANGE BASE
$SL(3,2)$	$2^3 \cdot 3 \cdot 7$	3	2	G	$2^3 3 \cdot 7$	1	0.83	0.39
$Sp(4,4)$	$2^8 \cdot 3^2 \cdot 5^2 \cdot 17$	4	4	Σ_3	$2 \cdot 3$	$2^2 \cdot 3 \cdot 5$	3.68	0.93
				Σ_4	$2^3 \cdot 3$	2^2	5.84	1.67
				Σ_5	$2^3 \cdot 3 \cdot 5$	1	1.58	0.07
				$\Sigma_2 \times \Sigma_4$	$2^4 \cdot 3$	2^2	2.78	0.05
				Σ_6	$2^4 \cdot 3^2 \cdot 5$	1	1.12	0.71

TABLE V

Conjugacy of elements in permutation groups

G	$\|G\|$	$\|X\|$	TIME
M_{10}	$2^4 \cdot 3^2 \cdot 5$	10	0.15
M_{11}	$2^4 \cdot 3^2 \cdot 5 \cdot 11$	11	0.25
–	$2^9 \cdot 3^4$	12	0.31
–	2^{10}	16	0.38
M_{22}	$2^7 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11$	22	0.36
M_{24}	$2^{10} \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 23$	24	0.80
$Sz(8)$	$2^6 \cdot 5 \cdot 7 \cdot 13$	65	0.65

Department of Mathematics
McGill University
Montreal, Quebec, Canada H3A 2K6

Department of Computer Science
Concordia University
Montreal, Quebec, Canada H3G 1M8

1. GREGORY BUTLER, *Computational Approaches to Certain Problems in the Theory of Finite Groups*, Ph. D. Thesis, University of Sydney, 1979.

2. GREGORY BUTLER, "Computing normalizers in permutation groups," *J. Algorithms*. (To appear.)

3. GREGORY BUTLER & JOHN J. CANNON, "Computing in permutation and matrix groups. I: Normal closure, commutator subgroup, series," *Math. Comp.*, v.39, 1982, pp.

4. GREGORY BUTLER & JOHN J. CANNON, "Computing in permutation and matrix groups. III: Sylow subgroups." (Manuscript.)

5. JOHN J. CANNON, "Software tools for group theory," *Proc. Sympos. Pure Math.*, vol. 37, Amer. Math. Soc., Providence, R. I., 1980, pp. 495–502.

6. JOHN J. CANNON, ROBYN GALLAGHER & KIM MCALLISTER, "STACKHANDLER: A language extension for low level set processing," *Programming and Implementation Manual*, TR 5, Computer-Aided Mathematics Project, Department of Pure Mathematics, University of Sydney, 1974.

7. CHRISTOPH M. HOFFMAN, "On the complexity of intersecting permutation groups and its relationship with graph isomorphism." (Manuscript.)

8. JAMES F. HURLEY & ARUNAS RUDVALIS, "Finite simple groups," *Amer. Math. Monthly*, v. 84, 1977, pp. 693–714.

9. JEFFREY S. LEON, "An algorithm for computing the automorphism group of a Hadamard matrix," *J. Combin. Theory Ser. A*, v. 27, 1979, pp. 289–306.

10. JEFFREY S. LEON, personal communication.

11. BRENDAN D. MCKAY, "Computing automorphisms and canonical labelling of graphs," *Lecture Notes in Math.*, vol. 686, Springer-Verlag, Berlin and New York, 1978, pp. 223–232.

12. HEINRICH ROBERTZ, *Eine Methode zur Berechnung der Automorphismengruppe einer endliche Gruppe*, Diplomarbeit, R. W. T. H. Aachen, 1976.

13. CHARLES C. SIMS, "Determining the conjugacy classes of a permutation group," *Computers in Algebra and Number Theory* (Proc. Sympos. on Appl. Math., New York, 1970), G. Birkhoff and M. Hall, Jr. (eds.), SIAM–AMS Proceedings, vol. 4, Amer. Math. Soc., Providence, R. I., 1971.

14. CHARLES C. SIMS, "Computation with permutation groups," *Proc. Second Sympos. on Symbolic and Algebraic Manipulation* (Los Angeles, 1971), S. R. Petrick (ed.), A. C. M., New York, 1971.