Reduction of Group Constructions to Point Stabilizers

Gene Cooperman and Larry Finkelstein*

College of Computer Science
Northeastern University
360 Huntington Ave.
Boston, Mass. 02115

Eugene Luks†

Department of Computer Science
University of Oregon
Eugene, Oregon 94705

ABSTRACT

The construction of point stabilizer subgroups is a problem which has been studied intensively. [1, 4, 5, 10, 11, 12, 14] This work describes a general reduction of certain group constructions to the point stabilizer problem. Examples are given for the centralizer, the normal closure, and a restricted group intersection problem. For the normal closure problem, this work provides an alternative to current algorithms, which are limited by the need for repeated closures under conjugation. For the centralizer and restricted group intersection problems, one can use an existing point stabilizer sequence along with a recent base change algorithm [2] to avoid generating a new point stabilizer sequence. This reduces the time complexity by at least an order of magnitude. Algorithms and theoretical time estimates for the special case of a small base are also summarized. An implementation is in progress.

1. INTRODUCTION

A fundamental problem in computational group theory is the construction of the *point stabilizer sequence* for a permutation subgroup G acting on an n-element subset A relative to an ordering $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_n$ of A. This is the chain of subgroups

$$G = G^{(1)} \supseteq G^{(2)} \supseteq \ldots \supseteq G^{(n)} = \{1\}$$

where $G^{(i)} = G_{\alpha_1\alpha_2\ldots\alpha_{i-1}}$ is the *pointwise stabilizer* of the set $\{\alpha_1, \ldots, \alpha_{i-1}\}$, $1 \leq i \leq n$. Because of its importance, a great deal of effort has been spent on developing efficient algorithmic solutions to this problem. [1, 4, 5, 10, 11, 12, 14] A solution usually consists of a set S of generators for G with the property that

$$G^{(i)} = \langle S \cap G^{(i)} \rangle,\ 1 \leq i \leq n-1.$$

A generating set for G with these properties is called a *strong generating set* relative to α.

* Research partially supported by NSF Grant DCR-8603293.

† Research partially supported by NSF Grant DCR-8609491 and ONR Grant N00014-86-0419.

In this paper, we present a general result for computational group theory which shows how to transform the construction of several important permutation subgroups related to G to a direct computation of the point stabilizer sequence. This is obtained by creating a new permutation group action on the disjoint union of two copies of the underlying point set which clearly reveals the subgroups in question as point stabilizer groups. New methods for constructing a strong generating set can then be applied [1] to reduce the worst case asymptotic running time for these constructions. In certain instances, a *change of base* can be used instead, for which a recent base change algorithm [2] further reduces the running time of our method.

Before stating our main result, we review the subgroups in question. Given subgroups H and G of $Sym(A)$, the *centralizer of* H *in* G, denoted $C_G(H)$, is the subgroup of G consisting of all elements of G which commute with each element of H. We refer to $C_G(G)$ as the *center of* G and denote it $Z(G)$. G is said to *normalize* H if H is invariant under conjugation by each element of G. The *conjugate of* H *by* g (gHg^{-1}), is denoted H^g. The *normal closure of* H *under* G is the smallest normal subgroup of $\langle H, G\rangle$ which contains H and is denoted $\langle H^G \rangle$. In the case where $H \subseteq G$, this is just the *normal closure* of H in G, denoted $\mathrm{NCL}_G(H)$. Our main results may be summarized as follows:

Theorem 1. *Let G and H be permutation groups acting on an n-element set A.*

(i) Suppose that a strong generating set is known for G. Then a strong generating set for $Z(G)$ can be constructed in time $O(n^3)$.

(ii) The construction of the normal closure of H under G, $\langle H^G \rangle$, can be obtained in the same time bound as required for computing a strong generating set for G.

(iii) Assume that strong generating sets are known for G and H. If G normalizes H, then $G \cap H$ can be constructed in time $O(n^3)$.

The construction for Theorem 1(i) actually reduces to the construction given in Theorem 1(iii) by setting $H = C_{Sym(A)}(G)$. In this case, it makes use of the fact first shown in [6] (see also [8]) that if S is a generating set for G then a strong generating set for $C_{Sym(A)}(G)$ can be determined in time $O(|S|n^2)$. We present an alternative proof of this fact in section 4 which makes intrinsic use of the structure of G as a permutation group. The reduction of the normal closure problem to the direct computation of a strong generating set is related to but distinct from existing methods used to solve this problem. [1, 3] Interestingly enough, efficient computation of normal closure plays a key role in the $O(n^4 log^c(n))$ algorithm [1] for the computation of a strong generating set. The idea of reducing the centralizer problem to one of constructing a strong generating set was previously observed in [13], but the construction of a strong generating set for $Z(G)$ depended on a group acting on n^2 points. (The current construction uses a group acting on $2n$ points.)

In section 2, we describe the basic transformation which is used to prove Theorem 1. We are able to prove Theorem 1(ii), (iii) as a direct consequence of this transformation. However, substantially more work is needed in order to derive Theorem 1(i). Section 3 provides some background material on group membership data structures which is required for the discussion in section 4 of the algorithm to construct a strong generating set for $C_{Sym(A)}(G)$. Section 4 establishes an upper bound of $O(|S|n^2)$ for this computation, which suffices to prove Theorem 1(i). However, our method can be substantially improved by applying new data structures [4, 5] which have been used for implementing a strong generating test. These ideas are outlined in section 5 together with a different estimate for computing $C_{Sym(A)}(G)$ which yields a better time for Theorem 1(i) when G has a small base.

2. MAIN REDUCTION

Let G and H be distinct permutation groups acting on the set, A. Let $\bar{A} = A_1 \dot{\cup} A_2$, where A_1 and A_2 are copies of A. Thus D can be described as the subgroup of $Sym(\bar{A})$ consisting of all elements of the form $\{(g, g): g \in G\}$. Let $K = \langle D \cup 1 \times H \rangle \subseteq G \times G$. Since each element of K can be written uniquely in the form (g_1, g_2) for suitably chosen elements $g_1 \in G$, and $g_2 \in \langle H, G \rangle$, we can define projection maps $f_1: K \to G$, and $f_2: K \to \langle H, G \rangle$ according to the rules $f_1((g_1, g_2)) \to g_1$ and $f_2((g_1, g_2)) \to g_2$. Finally, let $K_1 = f_1(K_{A_2})$ and $K_2 = f_2(K_{A_1})$ where K_{A_1} and K_{A_2} are the pointwise stabilizers subgroups in K of A_1 and A_2 respectively. To fix notation, for elements x, y of a group we refer to xyx^{-1} as the *conjugate* of y by x and denote it by y^x.

Theorem 2.1. *Under the above conditions, $K_1 = \langle H^G \rangle \cap G$ and $K_2 = \langle H^G \rangle$.*

Proof. Observe that an arbitrary element of K can be expressed in the form $\bar{g}_1 \bar{h}_1 \bar{g}_2 \bar{h}_2 \cdots \bar{g}_n \bar{h}_n$ for some n and $\bar{g}_1, \cdots, \bar{g}_n \in D$, $\bar{h}_1, \cdots, \bar{h}_n \in 1 \times H$. Letting $\bar{g}_i = (g_i, g_i)$ and $\bar{h}_i = (1, h_i)$ (1 is the identity), it follows that

$$\begin{aligned} K &= \{(g_1 \cdots g_n g_{n+1},\ g_1 h_1 g_2 h_2 \cdots g_n h_n g_{n+1})\} \\ &= \{(g_1 \cdots g_n g_{n+1},\ h_1^{g_1} h_2^{g_1 g_2} \cdots h_n^{g_1 \cdots g_n} g_1 g_2 \cdots g_n g_{n+1})\}. \end{aligned}$$

The last equation is similar to the one used to prove Schreier's lemma ([7], Lemma 7.2.2). In light of this description of K, we can "solve for g_{n+1}" in order to find the following characterizations of K_1 and K_2.

$$\begin{aligned} K_1 &= \{g_1 \cdots g_{n+1} \mid h_1^{g_1} h_2^{g_1 g_2} \cdots h_n^{g_1 \cdots g_n} g_1 g_2 \cdots g_n g_{n+1} = e\} \\ &= \{(h_1^{g_1} h_2^{g_1 g_2} \cdots h_n^{g_1 \cdots g_n})^{-1} \mid (h_1^{g_1} h_2^{g_1 g_2} \cdots h_n^{g_1 \cdots g_n})^{-1} \\ &\qquad\qquad = g_1 \cdots g_{n+1} \in G\} \\ &= \langle H^G \rangle \cap G \end{aligned}$$

$$\begin{aligned} K_2 &= \{h_1^{g_1} h_2^{g_1 g_2} \cdots h_n^{g_1 \cdots g_n} g_1 g_2 \cdots g_n g_{n+1} \mid g_1 \cdots g_{n+1} = e\} \\ &= \{h_1^{g_1} h_2^{g_1 g_2} \cdots h_n^{g_1 \cdots g_n} \mid g_1, \ldots, g_n, h_1, \ldots, h_n \text{ are} \\ &\qquad\qquad \text{all arbitrary}\} \\ &= \langle H^G \rangle \end{aligned}$$

The next result is important for Corollary 2.3. Its proof is clear.

Lemma 2.2.

(i) $K = D(1 \times H)$ if and only if $G \subseteq N_{Sym(A)}(H)$. (The first formula says K is factorizable.)

(ii) Let $a_1^{(1)}, \ldots, a_n^{(1)}$ be an ordering for G acting on A_1, and let $a_1^{(2)}, \ldots, a_n^{(2)}$ be the corresponding ordering for G acting on A_2. Suppose that $K = D(1 \times H)$. Then strong generators for K relative to the ordering $(a_1^{(1)}, \ldots, a_n^{(1)}, a_1^{(2)}, \ldots, a_n^{(2)})$ can be determined from strong generators for G and H. In fact, $K_{\{a_1^{(1)}, \ldots, a_i^{(1)}\}} = D_{\{a_1^{(1)}, \ldots, a_i^{(1)}\}}(1 \times H)$, and $K_{\{a_1^{(1)}, \ldots, a_n^{(1)}, a_1^{(2)}, \ldots, a_i^{(2)}\}} = (1 \times H_{\{a_1^{(2)}, \ldots, a_i^{(2)}\}})$, $1 \leq i \leq n$.

Corollary 2.3.

(i) If $H \subseteq G$, then $K_1 = K_2 = \mathrm{NCL}_G(H)$.

(ii) If G normalizes H, then $K_1 = G \cap H$ and $K_2 = H$. Given strong generators for G and H, strong generators for $G \cap H$ can then be computed in time $O(n^3)$.

Proof. The proof of (i) is obvious. To prove (ii), note that a strong generating set for K relative to the ordering

$$a_1^{(1)}, \ldots, a_n^{(1)}, a_1^{(2)}, \ldots, a_n^{(2)}$$

can be computed directly using part (ii) of the Lemma. Thus, $K_1 = G \cap H$ can be computed in time $O(n^3)$ [2] by using a change of base (or ordering) to

$$a_1^{(2)}, \ldots, a_n^{(2)}, a_1^{(1)}, \ldots, a_n^{(1)}.$$

□

The proof of Theorem 1(ii),(iii) follows directly from Corollary 2.3. The proof of Theorem 1(i) will be given at the end of section 4.

3. BACKGROUND

Let G be a permutation group acting on the set A and let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ be an ordering of A. As defined in section 1, the point stabilizer sequence relative to α is the sequence of subgroups

$$G = G^{(1)} \supseteq G^{(2)} \supseteq \ldots \supseteq G^{(n)} = \{1\}$$

where $G^{(i)} = G_{\alpha_1\alpha_2\ldots\alpha_{i-1}}$.

For each i, $1 \leq i \leq n$, let U_i be complete set of coset representatives for $G^{(i+1)}$ in $G^{(i)}$. We will always assume that the identity belongs to U_i. The set

$$U = \cup_{i=1}^{n-1} U_i$$

is called a *complete family of coset representatives* for the point stabilizer sequence of G relative to α. The associated cosets will be referred to as the *cosets of the point stabilizer sequence.* Each set U_i is in a 1-1 correspondence with the points in the orbit of α_i under $G^{(i)}$, denoted $\Delta^{(i)}$, in the sense that each element of U_i maps α_i to a distinct point of $\Delta^{(i)}$.

Suppose now that $g \in Sym(A)$, and we want to test if $g \in G$. Set $h = g$. If ${\alpha_1}^h \notin \Delta^{(1)}$, then $g \notin G$, and the test will fail. Otherwise, there exists a unique element $g_1 \in U_1$, such that ${\alpha_1}^h = {\alpha_1}^{g_1}$ and we continue the test with h replaced by gg_1^{-1}. Eventually, we arrive at an index j such that $h = gg_1^{-1}\ldots g_{j-1}^{-1} \in U_{j-1}$ with $g_i \in U_i$ for $1 \leq i \leq j-1$, h fixes $\alpha_1, \alpha_2, \ldots, \alpha_{j-1}$, and either h is the identity element or ${\alpha_j}^h \notin \Delta^{(j)}$. If the latter occurs, then we conclude that $g \notin G$. Otherwise, $g \in G$ and we can write g in the unique factored form

$$g = g_{j-1}g_{j-2}\cdots g_1, \; g_i \in U_i, \; 1 \leq i \leq j-1.$$

Note that testing for membership using this chain of subgroups takes $O(n^2)$ time.

A data structure is a *group membership data structure* for a group G (relative to an ordering α), if there is a corresponding algorithm to calculate a subset of a complete family of coset representatives for the point stabilizer sequence of G. If this subset is the entire family of coset representatives for the point stabilizer sequence then we say the data structure is *complete.* Both labelled branchings [10] and Schreier vectors [14] are examples of group membership data structures. In the case of labelled branchings the algorithm to recover a coset representative takes time $O(n)$, whereas in the case of Schreier vectors the time is $O(n^2)$. In practice, the recovery of coset representatives using Schreier vectors takes time which is usually linear in n, although examples exist for which the worst case time of n^2 occurs. Schreier vectors have the further advantage of being more space efficient when G has a small strong generating set. From the point of view of proving Theorem 1(i), we will always assume that it takes $O(n)$ time to recover a coset representative.

Let $S^{(i)} = G^{(i)} \cap S$. A group membership data structure is *fully augmented* (relative to an ordering α) for a set of generators S of G, if for each $\beta \in {\alpha_i}^{\langle S^{(i)}\rangle}$, the corresponding algorithm will compute an element of $\langle S^{(i)}\rangle$ which moves α_i to β. In particular, if S is a strong generating set for G, then a group membership data structure fully augmented for G must be complete.

Given a generating set S, it is possible to compute a Schreier vector data structure fully augmented for S in time $O(|S|n + n\log n)$ and a labelled branching fully augmented for S in time $O(|S|n + n^2)$. [4]

The essential facts which we require for section 4 can now be summarized as follows.

Lemma 3.1. *Given a set of generators S for $G \subseteq Sym(A)$, a group membership data structure can be computed in time $O(|S|n + n^2)$ which is fully augmented for S and such that the corresponding algorithm can recover an appropriate coset representative in time $O(n)$.*

4. CENTRALIZERS IN S_N

In this section, we describe an algorithm for computing the centralizer in S_n of a permutation group acting on $A = \{1, 2, \ldots, n\}$ using $O(|S|n^2)$ time. This algorithm is substantially different from the one described in [6, 8] in the sense that it relies on the properties of G as a permutation group (primitivity, transitivity, etc.) rather than on the cycle structure of a set S of generating permutations for G. The time estimates are the same, but this new version appears to allow faster implementations, and makes possible theoretically faster algorithms when G has a small base. The issues of implementation and a small base are postponed until section 5, while we present, here, a high level description which suffices to establish Theorem 1(i).

Let $Z = C_{Sym(A)}(G)$. We begin by giving a high level structure theorem for Z. If $\mathcal{O}_1$ and $\mathcal{O}_2$ are two orbits of G, then we say that $\mathcal{O}_1$ is *G-equivalent* to $\mathcal{O}_2$, denoted $\mathcal{O}_1 \equiv_G \mathcal{O}_2$, if there exists a one-one and onto map $\phi: \mathcal{O}_1 \mapsto \mathcal{O}_2$, such that $\forall a \in \mathcal{O}_1, \forall g \in G, a^{\phi g} = a^{g\phi}$. In this case, we say that ϕ *induces the G-equivalence.* We write $G|\mathcal{O}$ for the action of G restricted to the orbit $\mathcal{O}$.

Let $\mathcal{C}$ be a G-equivalence class of orbits with $|\mathcal{C}| = k$. For each $\mathcal{O} \in \mathcal{C}$, let $Z_{\mathcal{O}} = C_{Sym(\mathcal{O})}(G|\mathcal{O})$. The $Z_{\mathcal{O}}$ are isomorphic to each other. $Z_{\mathcal{O}}$ can be viewed as a subgroup of $Sym(A)$ by trivially extending each element outside of $\mathcal{O}$. We can then associate with each class $\mathcal{C}$, a direct factor $Z_{\mathcal{C}}$ of Z which has the following simple structure. It is the split extension of a direct product of the subgroups $Z_{\mathcal{O}}$ for each $\mathcal{O} \in \mathcal{C}$ by the symmetric group S_k acting naturally on the k orbits of $\mathcal{C}$. $Z_{\mathcal{C}}$ in fact can be described as a *wreath product* (see [9] for a formal definition of wreath products).

Lemma 4.1. *Let $\mathcal{C}_1, \ldots, \mathcal{C}_t$ be the G-equivalence classes of orbits of G acting on A. Let $|\mathcal{C}_i| = k_i$, $1 \leq i \leq t$ and let $Z_{\mathcal{C}_i} = C_{Sym(\mathcal{O}_j)}(G|\mathcal{O}_j)$, for some $\mathcal{O}_j \in \mathcal{C}_i$. Then Z is isomorphic to the direct product of the wreath products $Z_{\mathcal{C}_i} \wr S_{k_i}$, $1 \leq i \leq t$.*

The proof of Lemma 4.1 is straightforward and is omitted. According to this result, the construction of an algorithm for computing $C_{Sym(A)}(G)$ reduces to classifying the orbits of G under G-equivalence, and then for each i, $1 \leq i \leq t$, computing generators for $C_{Sym(\mathcal{O}_j)}(G|\mathcal{O}_j)$ for a single representative $\mathcal{O}_j \in \mathcal{C}_i$, $1 \leq i \leq t$. Lemmas 4.2–4.4 show how to compute generators on a single orbit for different cases.

We first review some standard definitions. *G acts transitively on A* if $a^G = A$ for $a \in A$. G is *semiregular* if

$G_a = \{e\}$ for all $a \in A$. G is *regular* if G is semiregular and acts transitively. This last is equivalent to $|G| = |A|$. $B \subseteq A$ is a *block of imprimitivity* for G if for all $g \in G$, $B^g \cap B = \emptyset$ or $B^g \cap B = B$. G is *primitive* if $\{e\}$ and A are the only blocks of imprimitivity.

Lemma 4.2. *Suppose that G acts regularly on A. Then Z is regular. Moreover we can describe Z precisely as follows: Let $g_i \in G$ be the unique element of G which moves 1 to i. Then Z is the set of permutations $\bar{g}$ in 1-1 correspondence with $g \in G$ defined by setting $i^{\bar{g}} = 1^{g^{-1}g_i}$.*

The proof is by direct computation, noticing that for $h \in G$ and $j = i^h$, $g_j = g_i h$.

Lemma 4.3. *Let B be the set of points fixed by G_1, so that $1 \in B$. Assume that G acts transitively on B. Then Z is semiregular. B is stabilized by Z, and so $|Z| \leq |B|$. Furthermore, if G is transitive on A, then B is a block of imprimitivity for G.*

Proof: The first statement is proved in [9]. To prove that Z stabilizes B, we simply use the fact that B represents the set of points fixed by a subgroup G_1 of G which is centralized by Z. Finally, since $G_a = G_1$ for each $a \in B$, it follows that B is a block of imprimitivity for G. □

Lemma 4.4. *Let B be the set of points fixed by G_1, so that $1 \in B$. If G is transitive on A, then $|B| = |Z|$ and Z can be described exactly.*

Proof: If $|B| = 1$, then by Lemma 4.3, Z fixes 1. It then follows from the transitivity of G that Z fixes every point, and hence is trivial. On the other hand, if $B = A$ then the result follows from Lemma 4.2. Thus, we may assume that B is non-trivial. In this case, B is a block of imprimitivity for G.

Let $G_{(B)}$ be the set stabilizer of B in G and let $H = G_{(B)}|B$ be the image of $G_{(B)}$ acting on B. Then H acts regularly on B. To see this, first note that if $g \in G$ moves 1 to any other distinct point in B, then g must set stabilize B, since B is a block. Hence, $G_{(B)}$ acts transitively on B. Furthermore, $G_a = G_B$ for any $a \in B$ implies that $H_a = \{e\}$.

It now follows from Lemma 4.2, that $U = C_{Sym(B)}(H)$ has order $|B|$ and can be described exactly. By Lemma 4.3, it suffices to show that each $\mu \in U$ can be extended uniquely to an element of Z.

Fix $a \in B$. Usually, a will be taken to be 1. Let $\mu \in U$. For $c \in A - B$, let $\sigma_c \in G$ move a to c. Extend μ to $A - B$ so that

$$c^\mu = a^{\mu\sigma_c}.$$

If $\sigma'_c \in G$ also moves a to c, then $\sigma'_c\sigma_c^{-1}$ fixes a. Hence $\sigma'_c\sigma_c^{-1}$ set stabilizes B, since B is a block of imprimitivity. But the structure of H then implies that $\sigma'_c\sigma_c^{-1}$ acts trivially on B. Therefore, the definition of c^μ is independent of the choice of σ_c.

We now have to show that μ centralizes G, or

$$x^{g\mu} = x^{\mu g}, \ \forall x \in A, \ g \in G.$$

Let $c = a^{\sigma_x g}$. Then we may take $\sigma_c = \sigma_x g$. Thus,

$$x^{g\mu} = a^{\sigma_x g\mu} = c^\mu = a^{\mu\sigma_c} = a^{\mu\sigma_x g} = x^{\mu g}$$

as required. □

Lemma 4.5. *Suppose that $\mathcal{O}_1$ and $\mathcal{O}_2$ are two orbits of G acting on A with $|\mathcal{O}_1| = |\mathcal{O}_2|$. Let S be a generating set for G. Then it is possible to test in time $O(|S||\mathcal{O}_1|^2)$ whether $\mathcal{O}_1$ and $\mathcal{O}_2$ are G-equivalent, and if so, to construct an explicit map ϕ.*

Proof: Suppose that we try to explicitly construct such a map ϕ. Then ϕ has to satisfy the following property:

$$i^{g\phi} = i^{\phi g}.$$

Let a be a fixed point of $\mathcal{O}_1$ and suppose that we choose some $b \in \mathcal{O}_2$ and restrict ϕ to satisfy $b = a^\phi$. Then G acts transitively on $\mathcal{O}_1$ implies that the above condition uniquely determines ϕ. More precisely, if $g \in S$, then $a^{g\phi} = a^{\phi g} = b^g$. Thus for each b, ϕ can be determined in time $O(|S||\mathcal{O}_1|)$ using a transitive closure type argument. Once ϕ has been determined, it takes $O(|S||\mathcal{O}_1|)$ time to verify whether ϕ is an equivalence. Since there are $|\mathcal{O}_1|$ choices for b, the result follows. □

We can now describe our algorithm. Let $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_r$ be the orbits of G acting on A. Assume that the points of A are ordered so that the points in $\mathcal{O}_1$ come first, to be followed by those in $\mathcal{O}_2$, etc. Let T hold the strong generating set for $C_{Sym(A)}(G)$, with T initialized to NIL. We iterate on i from r down to 1. We compute $C_{Sym(\mathcal{O}_i)}(G|\mathcal{O}_i)$ using Lemma 4.4, and append these generators to T. We now attempt to find the largest j, if any, with $1 \leq j < i$ and satisfying $G|\mathcal{O}_i \equiv G|\mathcal{O}_j$. If such a j is found, then we construct a permutation equivalence ϕ between $G|\mathcal{O}_i$ and $G|\mathcal{O}_j$ using Lemma 4.5. We may assume that ϕ interchanges $\mathcal{O}_i$ with $\mathcal{O}_j$ and is trivial on the remainder of A. We then append ϕ to T. We record the fact that $C_{Sym(\mathcal{O}_j)}(G|\mathcal{O}_j)$ is known. It is useful to observe that generators for $C_{Sym(\mathcal{O}_j)}(G|\mathcal{O}_j)$ need not be added to T. In the end, T will hold a strong generating set for $C_{Sym(A)}(G)$.

It is easy to show that the generators accumulated in T form a strong generating set for $C_{Sym(A)}(G)$. Furthermore, it is also straightforward to show that $|T| \leq n - 1$. These facts will be formally stated after the pseudocode is presented.

Centralizer Algorithm *Input:* Generators S for a subgroup G of $Sym(A)$. *Output:* A strong generating set T for $C_{Sym(A)}(G)$. *Local Variables:* L, a boolean array of length r with each entry initialized to FALSE. $L[i]$ will be set to TRUE if $C_{Sym(\mathcal{O}_i)}(G|\mathcal{O}_i)$ has already been computed. $\mathcal{B}$ will be used to hold a group membership data structure for $G|\mathcal{O}_i$ with generating set S which is fully augmented for S as discussed in section 3. The supporting routine Build-Group-Data performs the necessary construction.

```
Set $T \leftarrow$ NIL
Foreach $i$, set $L[i] \leftarrow$ FALSE
Let $\mathcal{O}_1, \ldots, \mathcal{O}_r$ be the orbits of $G$
    [We can assume that the points are ordered
```

```
      according to these orbits.]
  For $i \leftarrow r$ down to 1 do
    Set $\mathcal{B} \leftarrow$ Build-Group-Data($S, \mathcal{O}_i$)
    If $L[i]$ = FALSE then
      Set $C \leftarrow$ Transitive-Centralizer($S, \mathcal{O}_i, \mathcal{B}$)
      Append $C$ to $T$
        [We assume that each element of $C$ is
          trivially extended to $A$.]
    For $j \leftarrow i-1$ downto 1 do
      Set $\phi \leftarrow$ Test-Equivalence($S, \mathcal{B}, \mathcal{O}_i, \mathcal{O}_j$)
      If $\phi \neq$ NIL then
        Append $\phi$ to $T$
        Set $L[i] \leftarrow$ TRUE
        Break to the outer for loop (next $i$)
  return($T$)
```

We now present pseudocode for the supporting routines Transitive-Centralizer and Test-Equivalence.

Transitive-Centralizer *Input:* An orbit Δ of G acting on A, generators $S' = S|\Delta$ for $G|\Delta$, and a group membership data structure $\mathcal{B}$ for the action of G on Δ. *Output:* Generators for $C_{Sym(\Delta)}(G)$.

We will assume for simplicity that the points of Δ are renumbered from 1 to $|\Delta|$.

```
  Let $B$ be the set of points fixed by $G_1$
    [This can be computed using Schreier
      generators for $G_1$.]
  If $B = \Delta$ then ($G$ is regular)
    Return generators for $C_{Sym(A)}(G)$ using Lemma 4.2
  If $B = \{1\}$ then
    Return NIL [see Lemma 4.3]
  If $|B| > 1$ then
    Let $T = \{\tau_a, a \in B: \tau_a \in G,\ 1^{\tau_a} = a\}$
      [$T$ can be computed using $\mathcal{B}$ and
        generates $H = G_{(B)}/G_B$.]
    Let $U$ be generators for $C_{Sym(B)}(H)$ determined
      from Lemma 4.2
    Extend each element of $U$ of $\mathcal{O}_i$ as in Lemma 4.4
      [This requires the use of $\mathcal{B}$. ]
    return $U$
```

Test-Equivalence *Input:* Generators S for a subgroup G of $Sym(A)$, a group membership data structure $\mathcal{B}$ for G and orbits $\mathcal{O}_1$ and $\mathcal{O}_2$ of G. *Output:* Either NIL, or a map ϕ which induces a G-equivalence between $\mathcal{O}_1$ and $\mathcal{O}_2$.

In the case where the return value is not NIL, we may extend ϕ to A so that ϕ interchanges $\mathcal{O}_1$ and $\mathcal{O}_2$ and is trivial outside $\mathcal{O}_1 \cup \mathcal{O}_2$.

```
  If $|\mathcal{O}_1| \neq |\mathcal{O}_2|$ then return(NIL)
  Let $a$ be the first point of $\mathcal{O}_1$
  Foreach $b \in \mathcal{O}_2$ do
    [For each $b$, define a new $\phi$.]
    Set $a^\phi \leftarrow b$
    Foreach $\ell \in \mathcal{O}_2$
      Set $\ell^\phi \leftarrow$ NIL
    Foreach $k \in \mathcal{O}_1$ do
      Let $g \in G$ move $a$ to $k$  [$g$ determined from $\mathcal{B}$ ]
      [Require that $a^{g\phi} = a^{\phi g}$.]
      Set $k^\phi \leftarrow b^g$
      If $(b^g)^\phi$ = NIL then $(b^g)^\phi \leftarrow k$
      Else break to outermost foreach (next $b$)
    [Check if $\phi$ is an equivalence.]
    Foreach $g \in S$ do
      If $(g\phi)|\mathcal{O}_1 \neq (\phi g)|\mathcal{O}_1$ then
        break to outermost foreach (next $b$)
    Return($\phi$)
  Return(NIL)
```

Theorem 4.6. *The centralizer algorithm returns a strong generating set T consisting of at most $n-1$ generators in time $O(|S|n^2)$.*

Proof: The proof that the centralizer algorithm returns a strong generating set for $C_{Sym(A)}(G)$ is straightforward and relies on Lemma 4.1.

We turn our attention to the timing estimate for the algorithm. For simplicity, we may assume that each orbit $\mathcal{O}_1, \ldots, \mathcal{O}_r$ has the same size, so that $n = mr$. The computation of these orbits can be done in time $O(|S|n + n\log(n))$ using the procedure Augment in [4].

We will first show that each call to Transitive-Centralizer takes time $O(|S|n^2/r^2 + n^2/r)$. The computation of $\mathcal{B}$ takes time $O(|S|n^2/r^2)$. A simple way to obtain this result is to find a generating set for the point stabilizer and examine each generator in this set. We use the Schreier generators (see [7], Lemma 7.2.2). There are $O(|S|n/r)$ Schreier generators. The construction of each one takes time $O(n/r)$ since this involves the computation of two coset representatives (in time $O(n/r)$ by Lemma 3.1) and two multiplies restricted to the orbit. B is then computed by examining each Schreier generator, yielding a total time of $O(|S|n^2/r^2)$. The construction of the set T of generators for $G_{(B)}|B$ takes time $O(n^2/r^2)$), since this involves the construction of at most n/r coset representatives from $\mathcal{B}$ restricted to the orbit, and the computation of each one takes time $O(n/r)$, again by Lemma 3.1. Furthermore, the construction of U from T takes time $O(|T|n/r)$ and the extension of these elements to $\mathcal{O}_i$ and then to A takes time $O(|T|n)$. Since $|T| \leq n/r$, the result follows.

Since there are r orbits, the total time spent in calls to Transitive-Centralizer is $O(|S|n^2/r + n^2)$.

Now consider the time spent in calls to Test-Equivalence. In the worst case, there are $O(r^2)$ tests for G-equivalence between the r G-orbits. Each isomorphism test takes time $O(|S|n^2/r^2)$ by Lemma 4.5. Thus the total time spent in calls to Test-Equivalence is $O(|S|n^2)$.

Adding this up, we spend $O(|S|n^2/r + n^2)$ time in the calls to Transitive-Centralizer and $O(|S|n^2)$ time in the calls to Test-Equivalence. The only other additional work is the r calls to Build-Group-Data. By Lemma 3.1, each one of these can be done in time $O(|S|n/r + n^2/r^2)$. This contributes $O(|S|n + n^2/r)$ to the total running time. The accumulated time is thus $O(|S|n^2)$ as required.

To see that the algorithm returns at most $n-1$ generators, observe that at most $r-1$ isomorphisms between G-orbits are ever added to the generating set. Furthermore, at most $n/r - 1$ generators are added to the generating set from each G-orbit. Since there are r orbits, the size of the generating set returned is at most $r(n/r-1)+(r-1) = n-1$ as stated. □

Proof of Theorem 1(i): The result now follows directly by combining Corollary 2.3 with Theorem 4.6. $\square$

5. IMPLEMENTATION ISSUES

The base change algorithm is the bottleneck to faster algorithms. The ability to perform a base change on $\bar{A}$ in time $O(n^3)$ [2] is, as described in Corollary 2.3, the key step for obtaining upper bounds both for constructing a strong generating set for $Z(G)$ and, when G normalizes H, for constructing $H \cap G$. In the preceding sections, we have outlined algorithms which suffice for the proof of Theorem 1(i), (iii). In this section, we suggest two approaches which can be used to speed up these constructions. The approaches are being compared experimentally in an ongoing implementation.

The first approach relies on eliminating the $O(n)$ right cyclic shifts, each requiring $O(n^2)$ time, on which [2] relies. For our special case, we know that K is factorizable as $K = D(1 \times H)$. In this situation, we have developed a new base change algorithm requiring $O(m^2n^2)$ time, where both H and G have a small base of size m. To date, we have not been able to extend the new base change algorithm beyond this special case.

The second approach for improving the construction of $C_{Sym(\Omega)}(G)$ is to note from Theorem 4.6, that testing isomorphism of G-orbits dominates the running time. The following characterization can be used to simplify this part of the computation. The proof is clear, and is omitted.

Lemma 5.1. *If $a \in \mathcal{O}_i$, $i < j$, then $\mathcal{O}_i \equiv_G \mathcal{O}_j \iff G_a$ has a fixed point on $\mathcal{O}_j$.*

In order to exploit this, assume that the points of Ω are ordered so that the the orbits $\mathcal{O}_1, \ldots \mathcal{O}_r$ appear in consecutive order. We also assume assume that a strong generating set for G is known. This last fact is not required for the algorithm in Section 4, but is needed in order to apply Theorem 1. If α_1 is the first point of $\mathcal{O}_1$, then we know generators for G_{α_1} because of the existence of a strong generating set. In particular, we can compute the orbits of G_{α_1} on Ω and discover, via Lemma 5.1, precisely those orbits which are G-equivalent to $\mathcal{O}_1$.

In the case of orbits other than the first orbit, say $\mathcal{O}_j$, one doesn't readily have generators available for G_{α_j}, where $\alpha_j \in \mathcal{O}_j$. This situation can be easily remedied by doing a single right cyclic shift base change in $O(n^2)$ time so that α_j is now the first point in the ordering. Alternatively, a variation of the standard base change in [2] can be used to find fixed points of G_{α_j} without needing to output a new strong generating set. By a clever use of data structures similar in spirit to the algorithm for fully augmenting a group, [4] we can compute the fixed points of G_{α_j} on Ω in time
$O(|S|n/r + \min(\log|G|, n-1)n + \log|G| + n\log(n) + n^2/r)$, where it has been shown that $\log|G| \leq m\log(n)$.

Since this calculation must be carried out r times in the worst case, the total time spent checking G-equivalence of orbits in the centralizer algorithm can be given by
$O(|S|n + \min(\log|G|, n-1)rn + r\log|G| + rn\log(n) + n^2)$. The above time appears overly pessimistic in practice, since it assumes that most orbits are of a fixed size n/r, and the r orbits divide into $O(r)$ distinct G-equivalence classes. Thus we expect both theoretically and experimentally that our implementation using the above technique will be significantly faster than the $O(|S|n^2)$ reported for Test-Equivalence in section 4.

REFERENCES

1. L. Babai, E. Luks, and A. Seress, "On Managing Permutation Groups in $O(n^4 \log^c n)$", *Proc. 28th IEEE FOCS* (1988), 272–282.
2. C.A. Brown, L. Finkelstein, and P.W. Purdom, "A New Base Change Algorithm for Permutation Groups", *SIAM J. Computing*, to appear.
3. G. Butler and J.J. Cannon, "Computing in Permutation and Matrix Groups I: Normal Closure, Commutator Subgroups, Series" *Math. Comp.* **39** (1982), 663–670.
4. G. Cooperman and L.A. Finkelstein, "Short Presentations for Permutation Groups and a Strong Generating Test", submitted to J. Symbolic Computation.
5. G. Cooperman, L. Finkelstein and P. Purdom, "Fast Group Membership Using a Strong Generating Test", to appear in *Proc. of 1989 Computers and Mathematics Conference.*
6. M. Fontet, "Calcul du Centralisateur d'un Groupe de Permutations", *Bull. Soc. Math. France. Mémoires* **49–50**, (1977), 53–63.
7. M. Hall, Jr., *The Theory of Groups*, Macmillan, New York, 1959.
8. C.M. Hoffman, *Group-theoretic Algorithms and Graph Isomorphism*, Lecture Notes in Computer Science, **136**, Springer-Verlag, Berlin, 1982.
9. B. Huppert, "Endliche Gruppen I", Springer-Verlag, Berlin, 1967.
10. M. Jerrum, "A Compact Representation for Permutation Groups", *J. Algorithms* **7** (1986), 60–78.
11. D.E. Knuth, "Notes on Efficient Representation of Permutation Groups" (1981), unpublished manuscript.
12. J. Leon, "On an Algorithm for Finding a Base and Strong Generating Set for a Group Given by a Set of Generating Permutations", *Math. Comp.* **35** (1980), 941–974.
13. E.M. Luks, "Computing the Composition Factors of a Permutation Group in Polynomial Time", *Combinatorica* **7**, 87–99.
14. C.C. Sims, "Computation with Permutation Groups", in *Proc. Second Symposium on Symbolic and Algebraic Manipulation*, edited by S.R. Petrick, ACM, New York, 1971.