Nearly Linear Time Algorithms for Permutation Groups with a Small Base

László Babai[1,2*], Gene Cooperman[3†], Larry Finkelstein[3†], and Ákos Seress[4]

[1]Dept. of Comp. Science
University of Chicago
Chicago, Illinois 60637

[2]Dept. of Algebra
Eötvös University
Budapest, Hungary H-1088

[3]College of Comp. Science
Northeastern University
Boston, Mass. 02115

[4]Dept. of Mathematics
Ohio State University
Columbus, Ohio 43210

Abstract. A *base* of a permutation group G is a subset B of the permutation domain such that only the identity of G fixes B pointwise. The permutation representations of important classes of groups, including all finite simple groups other than the alternating groups, admit $O(\log n)$ size bases, where n is the size of the permutation domain. Groups with very small bases dominate the work on permutation groups in much of computational group theory. A series of new combinatorial results allows us to present Monte Carlo algorithms achieving $O(n \log^c n)$ (c a constant) time and space performance for such groups with respect to the fundamental operations of finding order and testing membership. (The input is a list of generators of the group.) Previous methods have achieved similar space performance only at the expense of increased time performance. Adaptations of a "cube-doubling" technique [BSz] and a local expansion property of groups [Ba3] (cf. [Ba4]) are the key to theoretically reducing the time complexity to $O(n \log^c n)$. The shared principal novelty of the new ideas is in their ability to build and manipulate certain chains of *subsets* of a group, which are not themselves subgroups, in order to build the point stabilizer subgroup chain. Further combinatorial ideas are used to lower the constant c. Comparative timing estimates, based on asymptotic worst-case analysis, lead us to expect a new implementation to be faster than previous implementations for groups of high degree.

1. INTRODUCTION

In his seminal work [Si], Sims introduced the notions of a *base* and *strong generating set* as the fundamental data structures for computing with permutation groups. We consider permutation groups given by a list of generators. A *base* for a permutation group G of degree n is a subset $B = \{\beta_1, \beta_2, .., \beta_M\}$ of the permutation domain, Ω, with the property that only the identity of G fixes each point of B. The point stabilizer sequence for G relative to B, considered as an ordered subset, is the chain of subgroups

$$G = G^{(1)} \geq G^{(2)} \geq \ldots \geq G^{(M+1)} = 1,$$

where $G^{(i)} = G_{\{\beta_1,..,\beta_{i-1}\}}$ is the subgroup of G which fixes pointwise the set $\{\beta_1, \ldots, \beta_{i-1}\}$, $1 \leq i \leq M+1$. A *strong generating set* (SGS) for G relative to B is a generating set S for G with the property that

$$\langle S \cap G^{(i)} \rangle = G^{(i)}, \text{ for } 1 \leq i \leq M+1.$$

Given B and S, it is straightforward to compute the order of G and test membership in G of an arbitrary element of S_n. This requires the construction, either implicitly as in [Je, Si] or explicitly as in [FHL, Kn], of a complete *transversal system* T for the point stabilizer sequence. T is the union of the transversals (complete sets of coset representatives) for $G^{(i+1)}$ in $G^{(i)}$, $1 \leq i \leq M$.

Let $\mu(G, \Omega)$ denote the minimum base size for a permutation group G. (Ω will be suppressed, where it is obvious.) If B is a *non-redundant base*, i.e. no proper subset of B is a base, then the following relation is immediate for $M = |B|$:

$$2^M \leq |G| \leq n^M. \qquad (*)$$

In fact, this inequality holds under the weaker and more easily verified assumption that B, as an ordered set, is nonredundant in the sense that no member of B is fixed by the pointwise stabilizer of the elements preceding it. The latter concept is called the *non-redundant base with respect to an ordering*, and is uniquely specified by an ordering of Ω. Blaha [Bl] has shown that a "greedy" ordering, choosing the next base point from the largest orbit of the pointwise stabilizer of the set of current base points, results in a nearly optimal base size $M = O(\mu(G) \log\log n)$.

Computational group theory algorithms whose complexity depends on the base size can be formulated naturally under three increasingly general hypotheses.

A. A possibly redundant base is known in advance.

B. An upper bound on the minimum base size $\mu(G)$ is known in advance.

C. Nothing is known in advance about bases or their sizes.

Section 2 provides the machinery to find an SGS under Hypothesis A in *deterministic* time $O(nM^c)$ where M is

*Research partially supported by NSF Grants CCR-8710078 and CCR-9014562.

†Research partially supported by NSF Grant CCR-8903952.

the size of a given base B. In section 3, a simple Monte Carlo solution is given in the general case (Hypothesis C) in time $O(n\mu(G)^c)$ (where ($\mu(G)$ is not assumed to be known). An asymptotically more efficient version follows in section 6 (with improved c).

The version in section 6 is suggested for implementations under Hypotheses B or C. The approach in section 6 starts with a solution under Hypothesis B. In addition, a very efficient strong generating test is described. These two are then combined to yield a solution under Hypothesis C: we *pretend* to have some upper bound M on $\mu(G)$, test whether the output of the algorithm is indeed an SGS; if not, we double the assumed bound M and repeat.

A family $\mathcal{G}$ of permutation groups is a *small base family* if there is a constant c such that for any group $G \in \mathcal{G}$ of degree n, there is a base for G of size $M \leq \log^c n$. When the family is clear, we refer to an individual member as a *small base group*. By inequality (*), this is equivalent to the condition that $\log|G| \leq \log^{c'} n$ for some constant c'.

The small base groups are especially important because all permutation representations of degree n of non-alternating simple groups have $O(\log n)$ size bases. This follows from bounds due to Kantor [Ka], via Cameron [Ca, Theorem 6.1]. (One often has fewer than eight base points for groups acting on ten thousand or more points.)

From a practical point of view, small base groups are the only groups with which one can effectively compute in the case of "very large" n. Indeed, a strong generating set requires $\Omega(n\mu(G))$ storage.

Rubik's cube-type puzzles and more general permutation puzzles don't fall under the "small base" heading. Such puzzles allow incremental solution: one can construct group elements with small support, i.e. sequences of moves that end up moving a small number of pieces. As we show in section 5, transitive groups with a small base cannot have non-identity elements with small support.

Some of our algorithms will be *Monte Carlo*. Such algorithms use random bits along the way and are not guaranteed to yield the correct result; but they are *likely* to do so on *every* input.

We say that the *reliability* of the algorithm is $p(n)$ if the probability of producing a correct output is at least $p(n)$ for every input group of degree n. If we mention a Monte Carlo algorithm without specifying its reliability, it will always be understood that the reliability is at least 0.9. In some cases, a greater reliability (such as $1 - 1/n$) will be stated.

To amplify the reliability from say $1-\delta$ to $1-\epsilon$, one has to repeat the algorithm $O(\log(\delta/\epsilon))$ times and take majority vote. (ϵ is chosen by the user.)

Our main theoretical result follows.

Theorem 1.1. *Given a permutation group G of degree n generated by s generators, one can construct an SGS by a Monte Carlo algorithm running in time $O(ns\log^c|G|)$ for some small absolute constant c. (No estimate on the order of G or the base size is assumed to be known in advance.)*

In particular, for small base groups this means *nearly linear time*, i.e. $O(ns\log^{c'} n)$. The proof will be given in section 3. (We note that in the general case (arbitrary base size) [Si, Je, Kn] require $O(n^5)$ time to construct an SGS, while [BCFLS] does the same in Monte Carlo time $O(n^3\log^3 n)$ (assuming $s = O(n)$).)

Finer estimates are obtained in terms of several parameters. The following result (reproduced later as Theorem 6.2) displays more complicated looking bounds which are easily seen to subsume Theorem 1.1.

Theorem 1.2. *Let $G = \langle S\rangle \leq Sym(\Omega)$ be a group with $|\Omega| = n$, and suppose that b is the maximal size of a non-redundant base. Then there exists a Monte Carlo algorithm which returns an SGS for G in $O(n\log^3|G| + nb^2\log^2|G|\log(b+\log n) + b^3(\log b)(\log^3|G|)(\log n)) + n|S|\log|G|)$ time. The constructed SGS supports membership testing in $O(n\log|G|)$ time and the memory requirement is $O(nb\log|G| + n|S|)$ during the construction.*

It should be stressed again that no prior bounds on base size or $|G|$ are required for this result (i.e it holds under Hypothesis C).

We note that usually $s = |S| = O(\log|G|)$ (this is certainly true if S is non-redundant, and often we have $s \leq 3$). Furthermore if $s = O(\log^2|G|)$ and $\mu(G) = O(\sqrt[3]{n}/\log n)$ (far weaker than the assumption of "small base"), the running time is dominated by the first term of the expression, $O(n\log^3|G|)$.

An important byproduct of these considerations is a very efficient Monte Carlo *strong generating test* for small base groups (fully stated as Theorem 6.1).

The basic approach is that of Sims and uses a variation of his Schreier vectors [Si], although portions of the algorithm were also influenced by Knuth's version [Kn]. There are, however, two quadratic bottlenecks in attempting to reduce the time of these traditional algorithms below $O(n^2)$. The first occurs in using $O(n)$ coset representatives to form all Schreier generators. If the coset representatives were calculated on all points, then $\Omega(n^2)$ time and space would be required. The second quadratic bottleneck occurs when sifting (also called factoring or stripping) the Schreier generators. The Schreier generators are a set of group elements, formed from generators of $G^{(i)}$ and coset representatives of the point stabilizer subgroup $G^{(i+1)}$ in $G^{(i)}$, that are guaranteed to generate $G^{(i+1)}$. Since there are at least as many Schreier generators as coset representatives, simply computing the action of $\Omega(n)$ Schreier generators on all n points would again require $\Omega(n^2)$ time.

Several radically new ideas that are *combinatorial* in nature were required in order to achieve these results. While our primary objective continues to be the construction of generators for certain *subgroups*, the shared principal novelty of the new ideas is in their ability to build and manipulate certain chains of *subsets* of a group, which are not themselves subgroups. Appropriately structured subsets allow a gradual buildup of the target subgroup by repeated doubling for which the familiar subgroup structure would be too coarse.

The solution to the first quadratic bottleneck is given in section 2 by a particularly efficient implementation of Sims's "Schreier vector" data structure which ensures time-efficient access to coset representatives without excessive storage requirement. This is accomplished through an intriguing algorithmic adaptation of a doubling technique due to Babai and

Szemerédi [BSz], originally developed for building (nondeterministically) short straight line programs in finite groups. This guarantees that the depth of the "Schreier trees" forming a Schreier vector is bounded by $\log |G|$.

The second quadratic bottleneck is eliminated in section 3 by showing that for small base groups, a "small" random number of Schreier generators suffice to generate the point stabilizer subgroup. Thus, only a "small" number of coset representatives need ever be computed on all n points. The technique is based on the "local expansion lemma" for groups [Ba3] (cf. [Ba2]), re-stated in Lemma 3.1. This result states that if S is a generating set for G closed under inverses and if D is a subset of G such that $|D| \leq |G|/2$ with the property that each element of D can be expressed as a word of length at most t in the elements of S, then there is an element $g \in S$ such that $|Dg \setminus D| \geq |D/4t|$. With this in place, a simple algorithm for constructing a strong generating set with complexity $O(n \log^c n)$ is given.

At least two other new combinatorial ideas are introduced in order to lower the power of $\log n$ in the complexity. One is to substitute short Schreier trees of depth $O(\log n)$ (as described in the paper of Cooperman, Finkelstein and Sarawagi [CFS]) for the cube Schreier trees of depth $O(\log |G|)$. The second is to use a combinatorial result showing that a group with a small base and a small number of orbits must have a large support. From this, one shows that a relatively small *random* subset of points is likely to intersect any undiscovered base points. (An implementation is still in progress.)

2. SCHREIER VECTOR DATA STRUCTURES AND CUBE SCHREIER TREES

In this section, we show how to construct especially efficient versions of Sims's Schreier vector data structure [Si]. These are important both for storing coset representatives of a point stabilizer subgroup chain, and as the basis of Sims's original group membership algorithm. The new version is shown both to be more space efficient and to enable faster computation of coset representatives, in the worst case. As such, they are the key to eliminating the first of the quadratic bottlenecks mentioned in the introduction. It also provides the machinery necessary for a $O(n \log^c n)$ deterministic group membership algorithm under Hypothesis A, although the algorithm is omitted due to lack of space.

Given a permutation group $G = \langle S \rangle \leq S_n$, a *Schreier vector data structure* for G is a sequence of pairs (R_i, T_i) called *Schreier trees*, one for each base point β_i, $1 \leq i \leq M$. T_i is a directed labelled tree, rooted at β_i, with edge labels from the set $R_i = \{g_1, \ldots, g_k\} \subseteq G^{(i)}$. The nodes of T_i are the points of the orbit $\beta_i^{\langle R_i \rangle}$. For a directed edge from u to v with label h, $u^h = v$. If v is a node of T_i, then the product of the edge labels along the path from β_i to v in T_i is a word in the elements of R_i whose corresponding permutation moves β_i to v. Thus each Schreier tree (R_i, T_i) defines a set of coset representatives for $G^{(i+1)}$ in $G^{(i)}$. The set of all such coset representatives forms a *partial transversal system* for G. A Schreier tree (R_i, T_i) is *complete* if $|T_i| = |\beta_i^{G^{(i)}}|$ and the Schreier vector data structure is *complete* if the trees are complete for each base point β_i. In this case, the Schreier vector data structure defines a *complete transversal system.*

Given $G \leq S_n$ and a permutation $g \in S_n$, a Schreier vector data structure for G is important for testing membership of g in G. For w a word on elements of G, if $w = g_1 g_2 \cdots g_k$ then $w^{-1} = g_k^{-1} \cdots g_2^{-1} g_1^{-1}$. The notation x^w denotes $x^{g_1 g_2 \cdots g_k}$, the image of x under the product of the elements of w. The time for computing the image of x is clearly $|w|$, the *length* of the word w. The *residue* of a permutation (or word) under the Schreier vector data structure is the word computed in the next procedure. Computing the *residue* (as a word, only) requires $O(rM)$ time, where r is the sum of depths of the trees in the Schreier vector data structure. (This assumes that for each $h \in \cup_j R_j$, both the permutations h and h^{-1} are stored.) We say that the permutation *sifts* if the product of the residue word, considered as a permutation, is the identity. A group element is in G if and only if it sifts. Testing the last property is known as a *group membership test.*

```
Procedure Residue-as-Word($g, \{(R_j, T_j) : 1 \leq j \leq M\}$)
Set $w \leftarrow g$ [if $g$ is permutation, $w$ is a word of length 1]
For $i \leftarrow 1$ to $M$ do
    Set $x = \beta_i^w$
    If $x$ is not a node of $T_i$ then return($w$)
    Let $w_i$ be the word consisting of the edge labels in $R_i$
        along the path from $\beta_i$ to $x$ in $T_i$
    Append $w_i^{-1}$ to the end of $w$
Return($w$)
```

The notion of a Schreier vector data structure was introduced by Sims in order to save up to a factor of n in space for typical cases, at the cost of up to a factor of n in time in computing coset representatives. Nevertheless, there are pathological examples of Schreier trees for which the worst case time occurs, and no savings in space. For example, if G is the symmetric group given by generators $(1\ 2), (2\ 3), \ldots, (n-1\ n)$, consider a Schreier tree which uses those labels for $G/G^{(2)}$. Such a tree will have depth $n-1$. The remainder of this section is concerned with how to build *shallow* Schreier trees that are guaranteed to avoid such pathology.

This paper will primarily be concerned with monotone Schreier trees. A Schreier tree (R, T) is *monotone* if R is an ordered set and the edge labels along the path from the root β_i to each vertex v is a word in $R^{-1} \cup R = (g_k^{-1}, \ldots, g_1^{-1}, g_1, \ldots, g_k)$ with strictly increasing indices. It is clear that given the labels R of a monotone tree, one can recover a monotone Schreier tree of depth at most $2|R|$ in time $O(n|R|)$. We will invoke such a procedure under the name monotone-tree(R). Two examples of monotone Schreier trees are introduced in this paper. This section presents one based on the "cube-doubling" of Babai and Szemerédi [BSz]. Section 4 presents "shallower" trees based on those used in the random base change algorithm of Cooperman, Finkelstein and Sarawagi [CFS].

Let $R = (g_1, \ldots, g_k)$ be a sequence of elements of a group G. The *cube* $C(R)$ is the set of group elements $\{g_1^{e_1} g_2^{e_2} \cdots g_k^{e_k} : e_i \in \{0,1\}\}$ and $C^{-1}(R) = \{g \in G : g^{-1} \in C(g_1, \ldots, g_k)\}$. The cube is *non-degenerate* if $|C(R)| = 2^k$.

The cubes were used in [BSz] to show (non-constructively) the existence of short straight line programs in finite groups. We give an efficient algorithmic adaptation of theier basic doubling technique.. A simple observation from [BSz] follows.

Proposition 2.1. *Let* $(g_1, \ldots, g_k, g_{k+1})$ *be a sequence of* *group elements and $C = C(g_1, \ldots, g_k)$. Then $|C(g_1, \ldots, g_{k+1})| = 2|C|$ if and only if $g_{k+1} \notin C^{-1}C$. In particular, $C(g_1, \ldots, g_{k+1})$ is non-degenerate if and only if $C(g_1, \ldots, g_k)$ is non-degenerate and $g_{k+1} \notin C^{-1}C$.*

Proof. Obvious from the definition. □

```
Procedure Double-the-Cube($(R,T)$, $g$, $\beta$)
Input: a monotone Schreier tree $(R,T)$,
      a group element $g$ and
      a root node $\beta$
Output: either the original Schreier tree $(R,T)$ or a new
      monotone Schreier tree $(R',T')$ with $g \in \langle R' \rangle$ and
      $|C(R')| = 2|C(R)|$
Time: $O(n\, depth(T))$
If $\beta^{C^{-1}(R)C(R)g} \neq \beta^{C^{-1}(R)C(R)}$ then
      Let $g' \in C^{-1}(R)C(R)g \setminus C^{-1}(R)C(R)$ such that
            $\beta^{g'} \notin \beta^{C^{-1}(R)C(R)}$
      Set $R \leftarrow$ append($R, \{g'\}$)
      Set $T \leftarrow$ monotone-tree($R$) [using $R^{-1} \cup R$]
Return $(R,T)$
```

The claimed output and times of the procedure are clear from Proposition 2.1 and the observation that $\beta^{g'} \notin \beta^{C^{-1}(R)C(R)}$ implies that $g' \notin C^{-1}(R)C(R)$.

Lemma 2.2. *Let $G = \langle S \rangle \leq Sym(\Omega)$, $|\Omega| = n$, and x an arbitrary point of Ω. Then a monotone Schreier tree (R,T) with $|R| \leq \log|G|$ (and, consequently, T with depth at most $2\log|G|$) can be built in $O(n \log^2 |G| + n|S|)$ time.*

Proof. First, determine x^G in $O(n|S|)$ time. Set (R,T) to the trivial Schreier tree rooted at x. While there is a $g \in S$ with $x^{C^{-1}(R)C(R)g} \neq x^{C^{-1}(R)C(R)}$, execute `Double-the-Cube`($(R,T)$, g, x). Since $C(R) \subseteq G$ doubles in size at each step by Proposition 2.1, there can be at most $\log|G|$ iterations. Setting $(R,T) \leftarrow$ `monotone-tree`(R) then yields the result. □

3. RANDOM SCHREIER GENERATORS, RANDOM SUBPRODUCTS, AND A SIMPLE ALGORITHM

In general, there can be as many as $|S|n = \Omega(n)$ Schreier generators, and each Schreier generator costs $\Omega(n)$ to construct since a full permutation multiply must be performed. This is the second of the quadratic bottlenecks mentioned in the introduction. Under Hypothesis A (known base) one can initially evaluate Schreier generators as words only on a known set of base points in order to construct a strong generating set in $O(n \log^c n)$ time, but additional machinery is necessary when a base is not known in advance (Hypothesis B or C). The additional machinery requires Monte Carlo algorithms.

Typically, we have a set S of generators for G, subgroup $H \leq G_x$ for some $x \in \Omega$, and an SGS, S' for H. We want to test if $H = G_x$. The standard method is to sift all Schreier generators formed from S and a transversal for G_x in G through a complete transversal system for H formed using S'.

This section takes the approach of defining an appropriate "small" random subset of the Schreier generators which suffices to generate a point stabilizer subgroup with high confidence. Since this set will always be poly-logarithmic in size, one can evaluate them on all n points of the permutation domain, while avoiding a quadratic bottleneck. Using cube Schreier trees and a random set of Schreier generators then allows one to find an SGS under Hypothesis C in $O(n \log^c n)$ time for small base groups. Nevertheless, a sufficiently high constant c would be required, that it is unclear whether the method would be faster than existing $\Omega(n^2)$ algorithms when tested on groups within the range of current computers. Hence, after prescribing how large a set of random Schreier generators is required, the remainder of this section describes the method of random subproducts for improving the complexity of forming random Schreier generators. This is followed by two further sections improving additional aspects of the basic algorithm, and culminating in a solution under Hypothesis C with considerably lower asymptotic bounds.

3.1. RANDOM SCHREIER GENERATORS

Let Σ be the set of coset representatives of G mod G_x determined from T. For arbitrary $h \in G$, let $\overline{h}$ be the unique element of Σ such that $x^{\overline{h}} = x^h$. The *Schreier generators* for G_x are $\{\sigma g \overline{\sigma g}^{-1} : \sigma \in \Sigma, g \in S\}$. As is well known, the Schreier generators generate the point stabilizer subgroup G_x [Ha, Lemma 6.2.2].

The next lemma, from [Ba3, Lemma 10.2], is the key to knowing how many random Schreier generators suffice to generate a point stabilizer subgroup.

Lemma 3.1. *Let S denote a set of generators of the group G and set $T = S \cup S^{-1} \cup \{1\}$. Let D be any finite subset of T^t, the set of t-term products of members of T (in any order), such that $|D| \leq |G|/2$. Then for at least one generator $g \in S$, $|Dg \setminus D| \geq |D|/(4t)$.*

The application to our problem follows.

Lemma 3.2. *Let $H < \widetilde{G} \leq G$ (H a proper subgroup). Suppose that (R_j, T_j) are Schreier trees for $H^{(j)}/H^{(j+1)}$ for $1 \leq j \leq M$. Let U be a set of coset representatives determined by a complete Schreier tree $(\widetilde{S}, \widetilde{T})$ of $\widetilde{G}$ in G such that $S := \widetilde{S} \cup (\cup_{1 \leq j \leq M} R_j)$ generates G. Then there exists a $g \in S$ such that for a random $u \in U$, the Schreier generator $ug\overline{ug}^{-1} \in \widetilde{G} \setminus H$ ($\overline{ug} \in U$ defined as above) with probability at least $1/(4\, depth(\widetilde{T}) + 4 \sum_{1 \leq j \leq M} depth(T_j))$.*

Proof. Let $D = HU$. Since H is a subgroup of $\widetilde{G}$, $|H| \leq |\widetilde{G}|/2$, and so $|D| = |H||U| \leq |\widetilde{G}||U|/2 = |G|/2$. Hence we can apply Lemma 3.1 with $T = S \cup S^{-1} \cup \{1\}$, $t = depth(\widetilde{T}) + \sum_{1 \leq j \leq M} depth(T_j)$. Therefore, there exists a generator $g \in S$ such that for a random element $d = hu \in D$, $h \in H$, $u \in U$, with probability at least $1/(4t)$, $dg \notin D$. But $dg \in D$ if and only if $hug = hug\overline{ug}^{-1}\overline{ug} \in HU$. This is true if and only if $ug\overline{ug}^{-1}\overline{ug} \in HU$. Since $\widetilde{G}\overline{ug} \cap HU = H\overline{ug}$, it then follows that $dg \in D$ if and only if $ug\overline{ug}^{-1} \in H$. The test for $dg \notin D$ depended only on the random choice of u and not of h, completing the proof. □

3.2. THE CORE ALGORITHM

The ideas developed up to this point suffice for constructing an SGS in $O(n \log^c |G|)$ Monte Carlo time under

Hypothesis C. The algorithm presented here provides a simplified framework for later improvements. We shall refer to an ordering $(\beta_1, \ldots, \beta_n)$ of Ω.

```
Procedure Construct-SGS($S$)
*Input:* generators $S$ for $G$
*Output:* a strong generating set $\cup_k R_k$
*Reliability:* $1 - O(1/n^c)$
    Set $R_1 \leftarrow S$
    Set $T_1 \leftarrow$ monotone-tree($R_1$)
    If orbit of $\beta_1 \neq$ vertices of $T_1$ then
        Augment $(R_1, T_1)$ as in Lemma 2.2
        Until orbit of $\beta_1$ = vertices of $T_1$
    Complete-Point-Stabilizer-Subgroup-Simple($e$, 1)
    Return $\{(R_j, T_j)\}$
```

```
Procedure
    Complete-Point-Stabilizer-Subgroup-Simple($g$, $j$)
*Input:* group element $g$, point $j$
    such that $\beta_j \leq$ first-point-moved($g$)
*Global Variables:* Schreier vector data structure
    $\{(R_j, T_j) : j \geq 1\}$ and
    an ordering of the set $\Omega = (\beta_1, \beta_2, \ldots, \beta_n)$
*Output:* modified global data structure,
    $\{(R_j, T_j) : 1 \leq j \leq n-1\}$
*Reliability:* with probability $1 - O(1/n^c)$, finds an element
    of $(\langle \cup_{j \leq k} R_k \rangle)_{\beta_j} \setminus \langle \cup_{j+1 \leq k} R_k \rangle$ if it exists

[Note $\beta_j^{C^{-1}(R_j)C(R_j)}$ = nodes of $T_j$]
If $\beta_j^{C^{-1}(R_j)C(R_j)g} \neq \beta_j^{C^{-1}(R_j)C(R_j)}$ then
    Set $R_j \leftarrow R_j \cup \{g\}$
    While $\exists h \in \cup_{j \leq k \leq |B|} R_k$
        such that $\beta_j^{C^{-1}(R_j)C(R_j)h} \neq \beta_j^{C^{-1}(R_j)C(R_j)}$ do
        Set $(R_j, T_j) \leftarrow$ Double-the-Cube($(R_j, T_j), h, \beta_j$)
Repeat
    Repeat $(4 \sum_{j \leq k \leq n-1} depth(T_k))\ |\cup_{j \leq k \leq n-1} R_k|$ times
        Set $u \leftarrow$ random coset representative in $(R_j, T_j)$
        Set $g \leftarrow$ random element of $\cup_{j \leq k} R_k$
        Set $u \leftarrow ug$
            [$u$ has probability $\Omega(1/(depth(T_j)\,|\cup_{j \leq k \leq M} R_k|))$
             of producing a new element of $G^{(j+1)}$.]
        Set $h \leftarrow$ Residue-as-Word($u$, $\{(R_k, T_k) : j \leq k \leq n-1\}$)
        Evaluate $h$ as a permutation
        If $h \neq$ *identity* then
            Set $j' \leftarrow$ first-point-moved($h$)
            For $k = j'$ downto $j$ do
                If $k = j'$ or $R_k \neq \emptyset$ then
                    Complete-Point-Stabilizer-
                    Subgroup-Simple($h$, $k$)
Until the last $c' \log n$ iterations of outer repeat loop
  yielded only trivial residues [$c$ approp. constant]
```

Lemma 3.2 can be applied to show correctness, with $G = \langle \cup_{j \leq k \leq n-1} R_k \rangle$, $\widetilde{G} = (\langle \cup_{j \leq k \leq n-1} R_k \rangle)_{\beta_j}$, and $H = \langle \cup_{j+1 \leq k \leq n-1} R_k \rangle$. Lemma 3.2 then says that there exists a $g \in \cup_{j \leq k \leq n-1} R_k$ for which $1/(4 \sum_{j \leq k \leq n-1} depth(T_k))$ of the coset representatives, u, for $\langle \cup_{j \leq k \leq n-1} R_k \rangle / (\langle \cup_{j \leq k \leq n-1} R_k \rangle)_{\beta_j}$ yield Schreier generators ug with non-trivial residue. Hence, the inner repeat loop of Complete-Point-Stabilizer-Subgroup-Simple will have probability $\Omega(1)$ of finding a Schreier generator with non-trivial residue, if one exists. At most $n \log |G|$ new Schreier generators can be added to the cubes $C(R_j)$ over the n levels over the life of the algorithm. Since $n \log(|G|) \leq n^2 \log n$, $log(n \log |G|) \leq 3 \log n$. Thus by choosing a suitable c' for the outer repeat loop termination criteria, ($c' \geq 4$ will do), the outer repeat loop has probability $\Omega(1 - 1/n)$ that it will not exit while there remains a Schreier generator with non-trivial residue.

The required time is clearly $O(n \log^c |G|)$, while the space is $O(nb \log |G|)$. The constant c is not derived here, since in the remaining sections, using additional combinatorial techniques, a faster algorithm (smaller power of c) is derived. In addition, a space-time tradeoff will be demonstrated, in which the newer algorithm can operate in $O(n \log |G|)$ space (similarly to Sims's algorithm) at the cost of an additional factor of $O(\log |G|)$ in time.

The previous discussion completes a sketch of the proof of Theorem 1.1 from the introduction.

3.3. RANDOM SUBPRODUCTS

Random subproducts were first defined in [BLS, section 6.2] and were used extensively in [BCFLS]. They serve here to accelerate the algorithm that follows.

Definition. *A random subproduct of a sequence of group elements $(g_1, g_2, \ldots, g_k)$ is an instance of a product $g_1^{e_1} g_2^{e_2} \cdots g_k^{e_k}$ in which the e_i are independent random variables uniformly distributed over $\{0, 1\}$.*

Lemma 3.4. *Let $H < \widetilde{G} \leq G$ be groups. Let $S = \{g_1, g_2, \ldots, g_k\}$ generate G, and let U be a transversal of $\widetilde{G}$ in G. Let $P(g, x)$ be the proposition that $|\{u \in U : ug\overline{ug}^{-1} \in \widetilde{G} \setminus H\}| \geq x$ ($\overline{ug} \in U$ as previously defined). Assume there is at least one generator $g \in S$ such that $P(g, r)$ for some r, $0 < r \leq |U|$. (Which g satisfies $P(g, r)$ is not known a priori.) Then we can form a word w of length at most k (by random methods) in time $O(k)$ such that $P(w, r/4)$ with probability at least $1/4$.*

Proof. The word w is chosen to be either w' or w_3 (defined below), each with probability $1/2$. Let $g = g_j$ be a randomly chosen generator satisfying $P(g, r)$. Let $w' = g_1'^{e_1} g_2'^{e_2} \cdots g_k'^{e_k}$ be a random subproduct, where $(g_1', g_2', \ldots, g_k')$ is a random re-ordering of $(g_1, g_2, \ldots, g_k)$. (e_i is 0 or 1 with uniform probability.) Then $w' = w_1 g_j^{e_j} w_2$. Without loss of generality, we will assume that $length(w_2) \leq length(w_1)$. The method of proof will be to form a word w_3 with the same properties and random distribution as w_2. We will then find probabilities for $P(w_1 g_j^{e_j} w_2, r/4)$ and $P(w_3, r/4)$.

We first assume the probability space to be restricted to the union of the set of events in which $length(w_2) < length(w_1)$ and half of the set of events in which $length(w_2) = length(w_1)$. (The half of the events for the latter class are chosen independently and randomly.) The complementary set of events is handled similarly at the end of the proof. Choose a random number ℓ' from a uniform distribution from 0 to $k-1$, and let $\ell = \lfloor \ell'/2 \rfloor$. Then, since the length of w_2 is uniformly distributed between 0 and $k-1$, the distribution of ℓ corresponds to the distribution of the length of w_2 given that either $length(w_2) < length(w_1)$ or $length(w_2) = length(w_1)$ and an independent random event of probability 1/2 occurs. Next choose w_3 as a random subproduct on a randomly chosen subset of S of size ℓ. Then set w to either $w' = w_1 g_j^{e_j} w_2$ or to w_3, each with probability 1/2.

Next, note that $\mathrm{Prob}(P(w_1 g_j^{e_j}, r/2) \mid \overline{P(w_2, r/4)}) \geq 1/2$, where $\overline{P(w_2, r/4)}$ denotes the negated statement. To see this, note that although the length of w_1 can be affected by w_2, e_j is independent of w_2. Thus, if $P(w_1, r/2)$, then with probability 1/2, $e_j = 0$ and the claim follows. If $\overline{P(w_1, r/2)}$, then with probability 1/2, $e_j = 1$. In that case, recalling $P(g_j, r)$ and the Schreier generator decomposition $u w_1 g_j \overline{u w_1 g_j}^{-1} = (u w_1 \overline{u w_1}^{-1})(\overline{u w_1}\, g_j\, \overline{u w_1 g_j}^{-1})$ for all $u \in U$ then yields the claim.

For fixed w_1, w_2, and e_j, if $P(w_1 g_j^{e_j}, r/2)$ and $\overline{P(w_2, r/4)}$, then $P(w_1 g_j^{e_j} w_2, r/4)$. This follows from observing that for each $u_1 \in U$, there is a unique $u_2 = \overline{u_1 w_1 g_j^{e_j}} \in U$ such that $u_1 w_1 g_j^{e_j} u_2^{-1} \in \widetilde{G}$ and that $u_1 w \overline{u_1 w}^{-1} = (u_1 w_1 g_j^{e_j} u_2^{-1})(u_2 w_2 \overline{u_1 w}^{-1}) \in \widetilde{G}$. If we define $p = \mathrm{Prob}(P(w_2, r/4))$ and use the fact that $\mathrm{Prob}(P(w_1 g_j^{e_j}, r/2) \mid \overline{P(w_2, r/4)}) \geq 1/2$, then we observe that $\mathrm{Prob}(w', r/4) = \mathrm{Prob}(P(w_1 g_j^{e_j} w_2, r/4)) \geq (1-p)/2$.

Also, $\mathrm{Prob}(P(w_3, r/4) \mid g_j \notin w_3$ (for w_3 considered as a word) $) = \mathrm{Prob}(P(w_2, r/4)) = p$ by the construction of w_3. Since w_3 is defined to be of length at most $k/2$, $\mathrm{Prob}(P(w_3, r/4)) \geq p/2$. Combining this with the previous probability, we see that $\mathrm{Prob}(w, r/4) = \mathrm{Prob}(w', r/4)/2 + \mathrm{Prob}(w_3, r/4)/2 \geq (1-p)/4 + p/4 = 1/4$.

Finally, we must consider the set of events complementary to the original assumption. The same method then shows that $\mathrm{Prob}(P(g_j^{e_j} w_2, r/2)) \geq 1/2$, and the remainder of the proof then follows by replacing $P(w_2, r/4)$ by $P(w_1, r/4)$ above. □

The next procedure describes how to exploit Lemmas 3.2 and 3.4 in order to reduce the number of Schreier generators that must be computed.

```
Procedure Random-Schreier-Generator-
    as-Word($(R_1, T_1)$, $\cup_{2 \leq j \leq M} R_j$)
Input: a monotone Schreier tree $(R_1, T_1)$ for $G/G_{\beta_1}$ with
    generating set $R = \cup_{1 \leq j \leq M} R_j$ for $G$ such that for
    $k \geq 2$ and $H = \langle \cup_{2 \leq j \leq M} R_j \rangle < G_{\beta_1}$, $R_k$ is the label
    set of a complete monotone Schreier tree $(R_k, T_k)$ for
    $H^{(k)}/H^{(k+1)}$.
Output: a Schreier generator as a word in $R$ which is in
    $G_{\beta_1} \setminus H$.
Reliability: $1 - 1/(64 \sum_{1 \leq j \leq M} depth(T_j))$
Time: $O(|R|)$
Choose at random a coset representative $u$ of $G_{\beta_1}$
  in $G$ computed from $(R_1, T_1)$
Choose random subword $w$ from $R$ according to Lemma 3.4
Return($uw\overline{uw}^{-1}$)
```

Proposition 3.5. *If H is a proper subgroup of G_{β_1}, then Random-Schreier-Generator-as-Word produces a word in $G_{\beta_1} \setminus H$ with the time and reliability as claimed.*

Proof. By Lemma 3.2, there exists a $g \in R = \cup_{1 \leq j \leq M} R_j$ such that for a randomly chosen coset representative u of $\widetilde{G} = G_{\beta_1}$ in G computed from (R_1, T_1), the probability is $1/(4 \sum_{1 \leq j \leq M} depth(T_j))$ that $ug\overline{ug}^{-1}$ lies outside of any subgroup H of G_{β_1}. (Here, (R_1, T_1) corresponds to $(\widetilde{S}, \widetilde{T})$ in the lemma.) Setting $r = [G : G_{\beta_1}]/(4 \sum_{1 \leq j \leq M} depth(T_j))$, the hypothesis $P(g, r)$ of Lemma 3.4 is now satisfied, and the result follows by multiplying $r/4$ by the probability 1/4 of Lemma 3.4. □

4. SHORT SCHREIER TREES

This section shows how to reduce the depth of the Schreier trees. Section 2 showed how to construct cube Schreier trees, which were monotone of depth $\log |G|$. This section constructs short Schreier trees for G_x in G, which are monotone of depth $\log n$, thus accelerating the final group membership algorithm. It assumes the existence of $O(\log n)$ random group elements of G.

The new Schreier trees depend on the availability of a set of $O(\log n)$ random group elements of G. Cooperman, Finkelstein and Sarawagi have shown that with $O(\log n)$ random group elements, one can build a monotone Schreier tree of depth $O(\log n)$ [CFS, Theorem 3.5]. (The original constant of the theorem, 44, was recently revised to 21 by a finer analysis.) The source of such random elements for $G^{(i)}$ will be knowledge of complete Schreier trees for $G^{(j)}/G^{(j+1)}$ for all $j \geq i$.

Lemma 4.1. *Let $[G : G_x] = n_1$. Then $21\delta \log n_1$ random elements of G (in any order) define a monotone Schreier tree for G/G_x with probability at least $1 - 2^{-\delta \ln n_1}$.*

For a Schreier tree (R_j, T_j), let $n_j = |\beta_j^{\langle R_j \rangle}|$ below. Note that if (R_j, T_j), $1 \leq j \leq M$ is a complete Schreier vector data structure for G, a random element of G can be computed by multiplying together M different random coset representatives, one for each j. Further, the cost of such a random element will be $O(n \sum_{1 \leq j \leq M} depth(T_j))$.

```
Procedure Shorten-Schreier-Tree
Input: a base $B = \{\beta_1, \beta_2, \ldots, \beta_M\}$ for $G$ with $\beta_1 = x$ and
    a complete Schreier vector data structure $(R_j, T_j)$,
    $1 \leq j \leq M$ for $G$ such that $depth(T_j) \leq 21 \log n_j$,
    $2 \leq j \leq M$ and $depth(T_1) \leq 2 \log |G|$
Output: a new Schreier vector data structure $(R'_1, T'_1)$ such
    that $\beta_1^{\langle R'_1 \rangle} = \beta_1^{\langle R_1 \rangle}$, $depth(T'_1) \leq 21 \log n_1$, and
    $|R'_1| \leq 21 \log n_1$.
Reliability: $1 - O(1/n)$
Time: $O(n \log |G| \log n)$
  Set $R'_1$ to an ordered set
   of $21 \log n_1$ random group elements
  [A product of random cosets, one from each $(R_j, T_j)$,
   is a random element]
  Form a monotone Schreier tree $(R'_1, T'_1)$ from $R'_1$
  If there is a node in $T_1$ not in $T'_1$, then return "failure"
  Return $(R_1, T_1)$
```

Lemma 4.2. *Procedure Shorten-Schreier-Tree performs correctly with the advertised time and reliability.*

Proof. Correctness and reliability of the algorithm are clear from Lemma 4.1. Under the assumptions on (R_j, T_j), the cost of generating a random element will be $O(n \sum_{1 \leq j \leq M} depth(T_j)) = O(n \log |G| + n \sum_{2 \leq j \leq M} n_j) =$

$O(n \log |G|)$. The cost of generating $21 \log n_1$ random elements then satisfies the required time. It also dominates the cost of building the Schreier tree and testing that $depth(T_1') \leq \log n_1$. □

5. SMALL BASE IMPLIES LARGE SUPPORT

Being able to "guess" a base for the group helps further to reduce the time. It means that a Schreier generator can be represented as a word, and calculations done only on the image of an assumed base, not on all n points. Only if it is decided to add the group element to the SGS, is it multiplied out as a permutation. Thus, the dominant time will be the time to multiply out as a permutation those group elements added to the SGS, and then update the Schreier vector data structure.

The idea is encompassed in the following lemma. For a permutation $g \in Sym(\Omega)$, the *support* of g is $supp(g) = \{x \in \Omega : x^g \neq x\}$.

Lemma 5.1. *Let G be transitive, b the size of a minimal base of G, and m the minimal size of support for non-identity elements of G. Then $bm \geq n = |\Omega|$.*

Proof. Let B be a minimal base for G and K the support of some $g \neq 1$ of minimal size. We define two hypergraphs on Ω with edge sets $\mathcal{B} = \{B^h : h \in G\}$ and $\mathcal{K} = \{K^h : h \in G\}$, respectively. Both hypergraphs are uniform and, since G is transitive, both are regular. Let $deg(\mathcal{B})$ and $deg(\mathcal{K})$ denote the valencies of these hypergraphs. Counting the number of pairs (x, B') with $x \in \Omega$ and $B' \in \mathcal{B}$, uniformity and regularity imply $deg(\mathcal{B})n = b|\mathcal{B}|$. Similarly, $deg(\mathcal{K})n = m|\mathcal{K}|$. Moreover, since the elements of $\mathcal{B}$ are bases for G and the elements of $\mathcal{K}$ are supports of group elements in $\{g^h : h \in G\}$, $B' \cap K' \neq \emptyset$ for all $B' \in \mathcal{B}, K' \in \mathcal{K}$. A simple counting argument yields $deg(\mathcal{B})deg(\mathcal{K})n = |\{(B', K', x) : B' \in \mathcal{B}, K' \in \mathcal{K}, x \in B' \cap K'\}| \geq |\mathcal{B}||\mathcal{K}|$. Substituting the values of $deg(\mathcal{B})$ and $deg(\mathcal{K})$, we obtain $bm \geq n$. □

6. FAST SGS CONSTRUCTION FOR AN UNKNOWN BASE

In this section, we present modifications of the procedures Construct-SGS and Complete-Point-Stabilizer-Subgroup, first presented in section 3, to exploit the full power of all the techniques developed in this paper. These modifications work for transitive groups under either Hypothesis B, when we know in advance an upper bound b on the size of a non-redundant base, or under the more general Hypotheses C, when we have no prior knowledge about the size of a base. We present two versions of the main algorithm, Construct-SGS-Hypothesis-B and Construct-SGS-Hypothesis-C. Extensions to the intransitive case are more complicated, and are therefore omitted due to lack of space.

We maintain three global variables. First, there is an ordered set $B = (\beta_1, \beta_2, \ldots, \beta_{|B|}) \subseteq \Omega$. Second, there is a Schreier vector data structure $\{(R_j, T_j) : 1 \leq j \leq |B|\}$, which will always have the property that $\sum_{1 \leq j \leq |B|} depth(T_j) = O(\log |G|)$, even when we don't know $\log(|G|)$. Third there is a specialized Schreier vector data structure, a set of cube Schreier trees $\{(\mathcal{R}_j, \mathcal{T}_j) : 1 \leq j \leq |B|\}$. The cube Schreier trees $\{(\mathcal{R}_j, \mathcal{T}_j)\}$ redundantly represent the same information as $\{(R_j, T_j)\}$, but are used to bound the amount of time spent building coset representatives for $\{G^{(j)}/G^{(j+1)}\}$. Complete-Point-Stabilizer-Subgroup is passed three parameters, g, j and M, where $g \in G^{(j)}$ and M is a bound on the current base size. The value of M is used in Complete-Point-Stabilizer-Subgroup to test if a word is equivalent to the identity permutation when it is already known to be trivial on B. This is accomplished by testing if the group element moves any point of a randomly chosen subset of Ω of size $M \log(M)$. If no point is moved and $M \geq b$, then we may apply Lemma 5.1, to certify with some measure of confidence that the element is trivial. In the case of Hypothesis B, one can determine a value b, the maximal size of any non-redundant base, and then set $M = b$. Otherwise, we have no prior knowledge of the value of b and require the use of a fast strong generating test to have confidence in the answer returned by Complete-Point-Stabilizer-Subgroup.

```
Procedure Construct-SGS-Hypothesis-B($S$,b)
Input: generators $S$ for $G$ and
    an upper bound $b$ on base size
Global Variables: An ordered set $B = (\beta_1, \beta_2, \ldots, \beta_{|B|}) \subseteq \Omega$,
    a Schreier vector data structure $\{(R_j, T_j) : j \geq 1\}$ with
    $\sum_{1 \leq j \leq |B|} depth(T_j) = O(\log |G|)$, and
    cube Schreier trees $\{(\mathcal{R}_j, \mathcal{T}_j) : j \geq 1\}$ such that
    $\beta_j^{C^{-1}(R_j)C(R_j)}$ = nodes of $\mathcal{T}_j$ on entry
Output: a strong generating set $\cup_k R_k$

Initialize $\{(R_j, T_j)\}$, $\{(\mathcal{R}_j, \mathcal{T}_j)\}$ to trivial trees
Set $B \leftarrow \{1\}$
   For each $g \in S$ do
      Set $h \leftarrow$ Residue-as-Word($g$)
      Evaluate $h$ as permutation
      If $h \neq identity$ then
         Set $j \leftarrow$ first-point-moved($h$)
         For $i \leftarrow j$ downto 1 do
            Complete-Point-Stabilizer-Subgroup($h, i, M$)
Return $\{(R_j, T_j)\}$
```

```
Procedure Construct-SGS-Hypothesis-C($S$, $M$)
Input: generators $S$ for $G$ and
    an upper bound $M$ on base size
Global Variables: An ordered set $B = (\beta_1, \beta_2, \ldots, \beta_{|B|}) \subseteq \Omega$,
    a Schreier vector data structure $\{(R_j, T_j) : j \geq 1\}$ with
    $\sum_{1 \leq j \leq |B|} depth(T_j) = O(\log |G|)$, and
    cube Schreier trees $\{(\mathcal{R}_j, \mathcal{T}_j) : j \geq 1\}$ such that
    $\beta_j^{C^{-1}(R_j)C(R_j)}$ = nodes of $\mathcal{T}_j$ on entry
Output: a strong generating set $\cup_k R_k$

Initialize $\{(R_j, T_j)\}$, $\{(\mathcal{R}_j, \mathcal{T}_j)\}$ to trivial trees
Set $B \leftarrow \{1\}$
   For each $g \in S$ do
      Set $h \leftarrow$ Residue-as-Word($g$)
      Evaluate $h$ as permutation
      If $h \neq identity$ then
         Set $j \leftarrow$ first-point-moved($h$)
         For $i \leftarrow j$ downto 1 do
            Sift:
            Complete-Point-Stabilizer-Subgroup($h, i, M$)
            If returned failure, then
               Set $M \leftarrow 2M$
               Goto "Sift"
Loop:
   For $i \leftarrow |B|$ downto 1 do
      Complete-Point-Stabilizer-Subgroup($e$, $i$, $M$)
      If Complete-Point-Stabilizer-Subgroup
         returned failure then goto "Failure"
```

```
  Repeat
    Do Strong-Generating-Test
    [described in Theorem 6.1]
    If Strong-Generating-Test returned failure then
      Goto "Failure"
  Until success of test for $\lceil \log\log n \rceil$ iterations
  Return $\{(R_j, T_j)\}$
  Failure:
    Set $M \leftarrow 2M$
    Goto "Loop"
```

```
Procedure
  Complete-Point-Stabilizer-Subgroup($g$, $j$, $M$)
Input: group element $g$, point $j$
    such that $\beta_j \leq$ first-point-moved($g$), $j \leq M$, and
    upper bound $M$ on base size
Global Variables: an ordered set $B = (\beta_1, \beta_2, \ldots, \beta_{|B|}) \subseteq \Omega$,
    Schreier vector data structure $\{(R_j, T_j) : j \geq 1\}$
    cube Schreier trees $\{(\mathcal{R}_j, \mathcal{T}_j) : j \geq 1\}$
    [The inequality $\sum_{1 \leq j \leq |B|} depth(T_j) = O(\log |G|)$ will
    always hold, and we shall have
    $\beta_j^{C^{-1}(R_j)C(R_j)} = \beta_j^{C^{-1}(R_j)C(R_j)}$ = nodes of $T_j$ on
    entry]
Output: modified global data structure,
    $\{(R_j, T_j) : 1 \leq j \leq |B|\}$
Reliability: fixed probability close to one (say $\geq 0.9$) of
    finding element of $((\cup_{j \leq k} R_k))_{\beta_j} \setminus \langle \cup_{j+1 \leq k} R_k \rangle$ if it
    exists

[If $g$ augments $(\mathcal{R}_j, \mathcal{T}_j)$, then find
 current coset representatives of $G^{(j)}/G^{(j+1)}$]
[Note $\beta_j^{C^{-1}(\mathcal{R}_j)C(\mathcal{R}_j)}$ = nodes of $T_j$]
If $\beta_j^{C^{-1}(\mathcal{R}_j)C(\mathcal{R}_j)g} \neq \beta_j^{C^{-1}(\mathcal{R}_j)C(\mathcal{R}_j)}$ then
  Set $\mathcal{R}_j \leftarrow \mathcal{R}_j \cup \{g\}$
  While $\exists h \in \cup_{j \leq k \leq |B|} \mathcal{R}_k$
      such that $\beta_j^{C^{-1}(\mathcal{R}_j)C(\mathcal{R}_j)h} \neq \beta_j^{C^{-1}(\mathcal{R}_j)C(\mathcal{R}_j)}$ do
    Set $(\mathcal{R}_j, \mathcal{T}_j) \leftarrow$ Double-the-Cube($(\mathcal{R}_j, \mathcal{T}_j)$, $h$, $\beta_j$)
  Set $(R_j, T_j) \leftarrow (\mathcal{R}_j, \mathcal{T}_j)$

[Add Schreier generators until, with high confidence,
 either $\cup_{j \leq k \leq |B|} R_k$ is a SGS or $B$ is not a base]
Repeat
  Repeat 64 $\sum_{j \leq k \leq |B|} depth(T_k)$ times
    [Random-Schreier-Generator-as-Word has
     probability $\Omega(1/(64 \sum_{j \leq k \leq |B|} depth(T_k)))$
     of producing a new element of $G^{(j+1)}$]
    Set $u \leftarrow$ Random-Schreier-Generator-
      as-Word($(R_j, T_j)$, $\cup_{j \leq k \leq |B|} R_k$)
    Set $h \leftarrow$ Residue-as-Word($u$, $\{(R_k, T_k) : j \leq k \leq |B|\}$)
    Let $B' \subseteq \Omega$ be a random subset of size $\min(M \log M, |\Omega|)$
    If $x^h \neq x$ for some $x \in B \cup B'$ then
      If $x^h = x$ for all $x \in B$ then [add base point]
        Set $\beta_{|B|+1} \leftarrow y \in B'$ such that $y^h \neq y$
        Set $B \leftarrow$ append($B$, $\{\beta_{|B|+1}\}$)
        Set $(R_{|B|+1}, T_{|B|+1})$ and $(\mathcal{R}_{|B|+1}, \mathcal{T}_{|B|+1})$
         to trivial Schreier tree
          [Set one root node, $\beta_{|B|+1}$, and no labels]
      Evaluate $h$ as a permutation
      Set $j' \leftarrow$ first-point-moved($h$)
      For $k = j'$ downto $j$ do
                    Complete-Point-Stabilizer-Subgroup($h$, $k$, $M$)
Until the last $c \log(n)$ iterations of outer repeat loop
  yielded only trivial residues on the random subsets
[$c$ is determined by the constants of the other routines]

[Ensure $depth(T_j) < O(21 \log n_j)$ after returning, to satisfy
 $\sum_{1 \leq j \leq |B|} depth(T_j) = O(\log |G|)$ (needed for timing)]
If $depth(T_j) > 2(21 \log n_j)$ or $|R_j| > 2(21 \log n_j)$ then
  Repeat
    Set $(R_j, T_j) \leftarrow$
      Shorten-Schreier-Tree($\{(R_k, T_k) : j \leq k \leq |B|\}$)
  Until Shorten-Schreier-Tree returns success once,
   or returns $\lceil \log(M \log |G| / \log n) / \log n \rceil$
   iterations with failure

[If failure, a base point is missing]
  If Shorten-Schreier-Tree returned failure, then
    Return to Construct-SGS with failure
```

Next we formalize our efficient *strong generating test* based on Lemma 3.1.

Theorem 6.1. *Assume we are given a candidate Schreier vector data structure $\{(R_j, T_j) : j \in B\}$ where B is a candidate base. Set $R = \bigcup_{1 \leq k \leq |B|} R_k$ and $D = \sum_{1 \leq k \leq |B|} depth(T_k)$. Then we can test whether or not R is an SGS for $G := \langle R \rangle$ in Monte Carlo time $O(n|B||R|D + n|S|D)$.*

Proof. The test consists of calling the inner repeat loop of Complete-Point-Stabilizer-Subgroup for each value of j from 1 to $|B|$ with $M = n$. (Computations are performed on all points of Ω.) The proof of Theorem 6.2 will demonstrate that at each level j, we will have probability $\Omega(1)$ of finding a new element of $\langle \cup_{j+1 \leq k \leq |B|} R_k \rangle$. The time will then be clear from the proof of Theorem 6.2. □

Remark. This algorithm has a worst-case space bound of $O(nb \log |G|)$, since Complete-Point-Stabilizer-Subgroup may recursively create several cube Schreier trees, each with $\Omega(\log |G|)$ distinct labels. Such space usage would be temporary, since just before returning from a level with such a tree, Complete-Point-Stabilizer-Subgroup would call Shorten-Schreier-Tree, yielding a Schreier tree of depth $O(\log n_j)$ and space usage $O(n \log n_j)$. However, initial experiments indicate that even this temporarily excessive usage of space would not usually occur. Even a single cube Schreier tree tends to have depth $O(\log n)$, and the chances are remote that the routine would simultaneously maintain several cube Schreier trees, each with $\Omega(\log |G|)$ distinct labels. For those, who are worried about theoretical worst cases, one can state a variant algorithm with worst case space bounds of $O(n \log |G|)$, but a worst case time bound that is a factor $O(\log |G|)$ worse than for the current algorithm.

Theorem 6.2. *Let $G = \langle S \rangle \leq Sym(\Omega)$ be a group with $|\Omega| = n$, and suppose that b is the maximal size of a non-redundant base. Then the Monte Carlo algorithm* Construct-SGS(S,1) *(either version B or C) returns an SGS for the case of transitive G in $O(n \log^3 |G| + b^3 (\log b)(\log^3 |G|)(\log n) + n|S| \log |G|)$ time. For intransitive G, an extension of the algorithm returns an SGS in $O(n \log^3 |G| + nb^2 \log^2 |G| \log(b + \log n) +$

$b^3(\log b)(\log^3|G|)(\log n) + n|S|\log|G|)$ *time. The constructed SGS supports membership testing in* $O(n\log|G|)$ *time and the memory requirement is* $O(nb\log|G| + n|S|)$ *during the construction.*

It should be noted that this routine may return an SGS with respect to an unpredictable ordering of Ω. However, a base change to a prescribed ordering can then be done in $O(n\log^2|G|)$ Monte Carlo time [CFS].

Proof. Due to the large number of details and lack of space, we only a *sketch* the proof for the *transitive case.* In particular we warn the reader that the pseudocode given earlier in this section applies to the transitive case only. An extension to the intransitive case with the same performance bounds will be described in an expanded version of this paper. It depends on carrying out the current algorithm with all base points restricted to the first orbit until a strong generating test restricted to the first orbit returns success. All global data structures are retained, and the algorithm is repeated on the union of the first first and second orbits, followed by the union of the first three orbits, etc.

It is crucial to observe that the algorithm goes through two phases: $M < b$ and $M \geq b$, where M is the currently assumed size of the base, and b is as in the theorem. Concerning issues of reliability, in the first phase it is only necessary to show that Strong-Generating-Test will return false with high probability if the strong generating set is not complete. This assures that M will continue to be doubled either until (by luck) we achieve a strong generating set, or until (more likely) $M \geq b$ and we enter the second phase. In the second phase ($M \geq b$), we will show that the calls to Complete-Point-Stabilizer-Subgroup construct a strong generating set before Strong-Generating-Test is called again.

A second overall issue is to find upper bounds on the number of times that Complete-Point-Stabilizer-Subgroup can be called with a non-trivial group element g. Whenever this occurs with parameter j, g must have been a new element of $\langle \cup_{\ell\leq k\leq|B|} R_k\rangle$ for some value $\ell \geq j$. Since the length of a chain of subgroups is at most $\log|G|$, Complete-Point-Stabilizer-Subgroup can be called at most $\log|G|$ times for each parameter j. Thus, the number of overall calls is bounded by $b\log|G|$.

Proof of Reliability: The reliability of the first phase relies on demonstrating the reliability of Strong-Generating-Test. Recall that this involves a call to a modified Complete-Point-Stabilizer-Subgroup with parameter $M = n$. The reliability of this routine is at least as high as that of a call to Complete-Point-Stabilizer-Subgroup with parameter $M \geq b$. Since the next part of the proof demonstrates that the reliability for $M \geq b$ is $\Omega(1)$, we assume the fact here. Further, when Strong-Generating-Test returns failure, it exhibits a certificate of failure so this conclusion will never be reached erroneously. Next, note that each repeat loop with $\lceil\log\log n\rceil$ iterations of Strong-Generating-Test can be called at most $\log b$ times with failure (and at most once with success) during the first phase ($M < b$), since each failure triggers a doubling of M. Thus, if each call to Strong-Generating-Test returning success is verified at least $\log\log b$ times, then we have $\Omega(1)$ confidence that over the life of the algorithm the $O(\log b)$ calls to the repeat loop of Strong-Generating-Test will never falsely report success.

Next, we consider the reliability of the second phase. Double-the-Cube is deterministic, and cannot fail. Shorten-Schreier-Tree is executed at most once per call to Complete-Point-Stabilizer-Subgroup, or at most $b\log|G|$ times. We can inductively assume that when Shorten-Schreier-Tree is called at level j in the second phase, $\cup_{j\leq k\leq|B|}R_k$ is a strong generating set for $\langle\cup_{j\leq k\leq|B|}R_k\rangle$. Since each call is attempted up to $\log(M\log|G|/\log n)/\log n$ times until success is achieved, when combined with the fact that Shorten-Schreier-Tree has reliability $1 - O(1/n)$, we can conclude that Shorten-Schreier-Tree has probability $\Omega(1)$ of never failing.

Next for parameter j, the inner repeat loop around Random-Schreier-Generator-as-Word has probability $\Omega(1)$ that a group element $h \notin \langle\cup_{j+1\leq k\leq|B|}R_k\rangle$ will be generated, if possible. This is seen since Random-Schreier-Generator-as-Word has probability $\Omega(1/(64\sum_{j\leq k\leq|B|} depth(T_k)))$ of generating such a group element, and it is repeated $64\sum_{j\leq k\leq M} depth(T_k)$ times. By an elementary probability argument using Lemma 5.1, when $M \geq b$, we have probability $\Omega(1)$ of generating a residue h, such that $supp(h)$ will intersect with $B\cup B'$. This will cause some residue, h, to be evaluated as a permutation, and added to the strong generating set with probability $\Omega(1)$.

Over the life of the algorithm $O(n\log|G|)$ such elements will be added to the strong generating set. Since the outer repeat loop executes up to $O(\log(n\log|G|)) = O(\log n)$ times, with reliability $\Omega(1)$ all required group elements for a strong generating set will be constructed.

Proof of Timing: We are now ready to analyze the timing. The cost of forming the cube Schreier trees $\{(\mathcal{R}_j, \mathcal{T}_j)\}$ is $O(nb\log^2|G|)$. The routine Double-the-Cube costs $O(n\log|G|)$. Over the whole algorithm, it can be called at most $\log|G|$ times per level, since each call must double the size of the cube. Multiplying the factors and the b levels yields the time for the cube Schreier trees.

The cost of the calls to Shorten-Schreier-Tree is $O(n\log^3|G|)$ since the cost of building the short Schreier tree is $O(n\log|G|\log n_j)$. There are at most $\lceil\log(b\log|G|/\log n)/\log n\rceil \leq \lceil 2\log b/\log n\rceil \leq 2$ calls to Shorten-Schreier-Tree for each call to Complete-Point-Stabilizer-Subgroup with non-trivial g. Since there are at most $O(\log|G|)$ calls per level (per j), the total time is $O(\sum_{1\leq j\leq M}(n\log|G|\log n_j)\log|G|) = O(n\log^3|G|)$.

The time for successful calls to Random-Schreier-Generator-as-Word and the associated calls to Residue-as-Word is $O(nb\log^2|G|)$. (We define a "successful" call as one that constructs a group element, u, whose residue is non-trivial on $B\cup B'$.) Note that the time for one such call is $O(n\log|G|)$. To see this, observe that $\sum_{j\leq k\leq|B|} depth(T_j) = O(\log|G|)$. Hence, h (generated from u) is a word of length $O(\log|G|)$ evaluated on n points. The number of successful calls is bounded by $O(b\log|G|)$, the number of calls to Complete-Point-Stabilizer-Subgroup. Hence, the overall time follows.

The time for unsuccessful calls to Random-Schreier-Generator-as-Word and the associated calls to Residue-as-Word is $O(b^3(\log b)(\log^3|G|)(\log n))$. In the proof of reliability, we showed that with probability $\Omega(1)$, $M < 2b$ on termination. So, the time for a single call is $O(M\log M\log|G|) = O(b(\log b)(\log|G|))$. There are at most $b(\log|G|)(\log n)$ unsuccessful calls before either a successful call is made, or before the strong generating test is called. There are at most $b \log |G|$ successful calls, since each one generates a call to `Complete-Point-Stabilizer-Subgroup`. There are at most $\log b$ calls to `Strong-Generating-Test`. Hence, the total number of unsuccessful calls is $O(b(\log |G|)(\log n) \cdot (b \log |G| + \log b)) = O(b^2 (\log^2 |G|)(\log n))$. (The unsuccessful calls made as part of the strong generating test are accounted for later, under the time for the strong generating test.) Combining this with the cost of a single unsuccessful call yields the overall stated time.

The cumulative time for the strong generating test is $O(nb \log^2 |G| \log b \log \log n)$. To see this, note that the cost of constructing a new random Schreier generator at level j and sifting it to obtain the residue is $n \log |G|$. As in the previous paragraph $O(\log |G|)$ random Schreier generators are required to assure constant probability of discovering non-trivial residue (where possible). The strong generating test can be called at most $\log b$ times since the assumed base size M doubles before each call, and the algorithm will find a strong generating set with high probability once $M \geq 2b$. The repeated calls to the strong generating test then cause an additional factor of $O(\log \log n)$ in the time. The sum of all the previous times then yields the theorem (for the transitive case). □

7. CONCLUSION.

A Monte Carlo algorithm for constructing a strong generating set in time $O(n \log^c n)$ for a base of size $O(\log^c n)$ was demonstrated in section 3. A more efficient version (smaller power c) was given in section 6. A deterministic construction of a strong generating set in time $O(n \log^c n)$ for a base of size $O(\log^c n)$ is also available based only on section 2, for the case of Hypothesis A (a known small base). (The formal algorithm was omitted for lack of space.)

The worst-case analysis of the general case (unknown base) frequently assumed that Schreier trees could be of depth $O(\log |G|)$, as seen for cube Schreier trees. Practical experience has shown that this theoretical worst case is highly unlikely. In a practical implementation, the new algorithm will be heuristically speeded up by using the Schreier trees of a traditional implementation (using breadth-first search), and reserving the cube and short Schreier trees as theoretical guarantees to be used only in the event that an unusually deep Schreier tree occurs.

While awaiting an implementation corresponding to Theorem 6.2, a tentative guess for the break-even point of the new algorithm can be made. For lack of a better estimate we take all asymptotic coefficients as 1, all logarithmic bases as 2, a Sims-type implementation is assumed to run in time $\Omega(n^2 b^2)$ in practice, and we assume the worst-case estimate of $O(nb \log^2 |G| \log b)$ for the new method. If we assume a base size of 10, the new method would be competitive for $n \geq 35{,}000$. Naturally, the true asymptotic coefficients of an implementation may yield a very different break-even point.

Acknowledgment. We are indebted to Gene Luks for inspiring discussions.

REFERENCES.

[Ba1] L. Babai, "Monte-Carlo Algorithms in Graph Isomorphism Testing", Université de Montréal Tech. Report D.M.S. 79-10 (1979), Dep. Math. et Stat.

[Ba2] L. Babai, "Local Expansion of Vertex-Transitive Graphs and Random Generation in Finite Groups", *Proc. 23rd ACM STOC* (1991), to appear.

[Ba3] L. Babai, "Bounded Round Interactive Proofs in Finite Groups", *SIAM J. Discr. Math.*, to appear.

[Ba4] L. Babai, "Complexity in Finite Groups", *Proc. International Congress of Mathematicians*, Kyoto, 1990, Springer-Verlag, to appear.

[BCFLS] L. Babai, G. Cooperman, L. Finkelstein, E.M. Luks, and Á. Seress, "Fast Monte Carlo Algorithms for Permutation Groups", *23rd ACM STOC* (1991), to app.

[BLS] L. Babai, E. Luks, and Á. Seress, "Fast Management of Permutation Groups", *Proc. 29th IEEE FOCS* (1988), pp. 272–282.

[BSz] L. Babai and E. Szemerédi, "On the Complexity of Matrix Group Problems I," *Proc. 25th IEEE FOCS* (1984), Palm Beach, FL, pp. 229–240.

[Bl] K. Blaha, "The Greedy Algorithm and Bases for Permutation Groups", *J. Algorithms*, to appear.

[BFP] C.A. Brown, L. Finkelstein, and P.W. Purdom, "A New Base Change Algorithm for Permutation Groups", *SIAM J. Computing* **18** (1989), pp. 1037–1047.

[Ca] P.J. Cameron, "Finite Permutation Groups and Finite Simple Groups", *Bull. London Math. Soc.*, **13**, 1981, pp. 1–22.

[Ch] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations", *Ann. Math. Stat.* **23**, (1952), pp. 493–507.

[Co] J.H. Conway et al., *Atlas of Finite Groups*, Clarendon Press, Oxford, 1985.

[CF] G. Cooperman and L. Finkelstein, "A Strong Generating Test and Short Presentations for Permutation Groups", to appear in *J. Symbolic Computation.*

[CFS] G. Cooperman, L. Finkelstein and N. Sarawagi, "A Random Base Change Algorithm for Permutation Groups", *Proc. Internat. Symp. on Symbolic and Algeb. Comput.*, Tokyo, 1990, ACM Press and Addison-Wesley, pp. 161–168.

[FHL] M. Furst, J. Hopcroft and E. Luks, "Polynomial Time Algorithms For Permutation Groups", *Proc. 21st IEEE FOCS* (1980), pp. 36–41.

[Ha] M. Hall, Jr., *The Theory of Groups*, Macmillan, New York, 1959.

[Je] M. Jerrum, "A Compact Representation for Permutation Groups", *J. Algorithms* **7** (1986), pp. 60–78.

[Ka] W.M. Kantor, "Permutation Representations of the Finite Classical Groups of Small Degree or Rank", *J. Algebra* **60** (1979), pp. 158–168.

[Kn] D.E. Knuth, "Notes on Efficient Representation of Perm Groups" *Combinatorica* **11** (1991), pp. 57–68 (preliminary version circulated since 1981).

[Si] C.C. Sims, "Computation with Permutation Groups", in *Proc. Second Symp. on Symbolic and Algeb. Manip.*, (S.R. Petrick, ed.), ACM, New York, 1971.