A Fast Cyclic Base Change for Permutation Groups

Gene Cooperman* and Larry Finkelstein*

College of Computer Science
Northeastern University
Boston, Mass. 02115

Abstract. Two new cyclic base change algorithms are presented for a permutation group G acting on n points. One is deterministic and the other is randomized. When G is a *small* base permutation group both algorithms have worst case time complexities which are better than existing algorithms in their class. The deterministic algorithm requires $O(n \log^2 |G| + n|S| \log |G|)$ time. It outputs a Schreier vector data structure which requires $O(n \log |G|)$ space and in which every Schreier tree has depth bounded by $2 \log |G|$. The randomized algorithm returns a Schreier vector data structure for which the sum of the depths of the resulting Schreier trees is $O(\log |G|)$. It is shown that the algorithm has probability exceeding $1 - 2/n$ of using $O(nb \log^2 n)$ time for b the size of a non-redundant base. As with most randomized base change algorithms, it is Las Vegas in the sense that within the same time it can be deterministically verified whether the answer is correct. In order to achieve this time bound it is necessary that random elements of G be computable in time $O(n \log |G|)$. A final result is a randomized algorithm which given an arbitrary strong generating set S for G constructs a Schreier vector data structure which can be used to compute random elements in $O(n \log |G|)$ time. It is shown that this algorithm has probability exceeding $1 - 1/|G|$ of using $O(n \log^2 |G| + n|S|)$ time.

1. INTRODUCTION

Let G be a permutation group acting on an n-element set Ω. Most important algorithms for performing computations with G assume the knowledge of a strong generating set, S, relative to some ordering α of Ω. Equally important is the computation of a new strong generating set for G relative to a different ordering α' of Ω, commonly referred to as a *base change*. An especially important subproblem occurs when α' is obtained from α by a *right cyclic shift*. This is called a *cyclic base change*. The main results of this papers are two new algorithms for performing a cyclic base change which are more time and space-efficient for the important class of *small base permutation groups* than existing algorithms. G is said to be a small base group with parameters c and d if $\log |G| \leq d \log^c |\Omega|$. For suitably chosen small constants c and d every permutation representation of a non-alternating simple group is a small base group.

Applications of the cyclic base change include search problems in the presence of symmetry [6, 7, 10, 16, 18], and a strong generating test, "verify", of Sims that has been implemented in the Cayley system [8]. Applications of the general base change include fast construction of strong generating sets for normal closure, center, and certain other subgroups [13], fast group membership [2, 3], and other structural problems [19].

The first result is a deterministic cyclic base change algorithm which is both space and time efficient.

Theorem A. *Given a strong generating set S for G, a deterministic cyclic base change algorithm can be described that requires $O(n \log^2 |G| + n|S| + n \log n)$ time and $O(n \log |G|)$ space*

Using randomized methods one can give a substantial theoretical improvement to Theorem A under the hypothesis of a *short Schreier vector data structure*. A formal definition is deferred, but such a data structure allows computation of random elements of G and sifting (or stripping or factoring) in $O(n \log |G|)$ time.

Theorem B. *Given a short Schreier vector data structure for G, a random cyclic base change algorithm can be described which has probability at least $1 - 2/n$ of using $O(nb \log^2 n)$ time and $O(nb \log n)$ space. Furthermore the algorithm returns a short Schreier vector data structure with respect to the new ordering*

The next result removes an unnecessary hypothesis in the general randomized base change algorithm given in [14] using techniques similar to those of Theorem A.

Theorem C. *Given a strong generating set S for G, an algorithm can be described for computing a short Schreier vector data structure which has probability at least $1 - 1/|G|$ of using $O(n \log^2 |G| + n|S| \log |G|)$ time*

An immediate corollary is the following stronger version of [14, Theorem B].

*Research partially supported by NSF Grant CCR-8903952.

ISSAC '92-7/92/CA, USA
© 1992 ACM 0-89791-490-2/92/0007/0224...$1.50

Corollary D. *Given a strong generating set S for G relative to an ordering α, an algorithm can be described for performing a base change relative to an arbitrary ordering α' using $O(n \log |G|)$ space and returning a short Schreier vector data structure with respect to the new ordering. The algorithm has probability at least $1-1/|G|$ of using $O(n \log^2 |G| + n|S|)$ time.*

In many applications, it is necessary to repeatedly perform cyclic base changes. The significance of Theorem B coupled with Theorem C is that after one pays an initial one-time cost for constructing a short Schreier vector data structure, the input and output data structures used for all subsequent base change computations will have this property.

The first base change algorithm was presented by Sims [19]. Sims's algorithm used a Schreier vector data structure in order to achieve the space efficiency required for working with permutation groups of very large degree and was shown to have worst case time complexity of $O(n^5)$ time. This time complexity can be refined by incorporating a parameter which measures the size of a *base* for G relative to the new ordering. Nevertheless, Sims's algorithm tends to work faster in practice than the worst case time complexities. The reason for this is that one must incorporate in the time estimates the depth of the Schreier trees which are used to implicitly represent the cosets for the subgroups of the point stabilizer sequence and which together form the Schreier vector data structure. In general, these depths tend to be "short", meaning $O(\log n)$, but unless care is taken, there is no guarantee that the trees won't have depth $\Theta(n)$.

Brown, Finkelstein and Purdom [5] presented a base change algorithm which has worst case time complexity $O(n^3)$ using a generalization of Sims's original argument and a new data structure for representing the cosets of the point stabilizer sequence, Jerrum's *labeled branching* [11, 15]. This result is derived from a $O(n^2)$ cyclic base change algorithm. Unfortunately, a labeled branching for G requires $\Theta(n^2)$ space when G is transitive on Ω, and so this data structure is only practical for moderate values of n.

Babai [1, Theorem 2.5] and Leon [17] were the first to apply randomized techniques to a base change. The current authors together with Namita Sarawagi [12, 14] used randomized methods to develop a general base change algorithm using $O(n \log^2 |G|)$ time and $O(n \log |G|)$ space, provided that random elements can be computed in $O(n \log |G|)$ time — a common situation. Thus, when the hypotheses are satisfied, the algorithm operates in $O(n \log^c n)$ time for a small base group. The output of this algorithm is always a short Schreier vector data structure allowing such fast computation of random elements, but the initial data structure potentially requires a more expensive pre-processing step. Theorem C of this paper does the pre-processing step in $O(n \log^2 |G| + n|S| \log |G|)$ time.

The key idea in [14] is to use randomized methods to build *short Schreier trees* with probabilistically guaranteed bounds on the depth of the Schreier trees. The alternative deterministic method of *cube Schreier trees* [3] was required in the development of an almost linear time group membership algorithm for small base permutation groups. It is based on an effective implementation of an idea due to Babai and Szemerédi for building straight line programs for finite groups [4] and produces Schreier trees of depth at most $2 \log |G|$. The main results of this paper represent a novel synthesis of the two methods of bounding the depth of Schreier trees.

Section 2 introduces notation, and the important constructions of short Schreier trees and cube Schreier trees. The proof of Theorem A is presented in section 3 as Theorem 3.1. Theorem B is given in section 4 as Theorem 4.1 and Theorem C in section 5 as Theorem 5.1. Most of the algorithmic subroutines described here have been implemented in the context of larger programs. For example, the routines used to compute short and cube Schreier trees play a crucial role in our implementation of the small base group membership algorithm [3]. Implementation of an independent base change algorithm based on the ideas described in this paper is currently undergoing testing. Based on our previous computational experience [12, 5], we expect to see a signficant improvement over existing methods.

2. SPACE-EFFICIENT GROUP MEMBERSHIP DATA STRUCTURES

A fundamental issue in computing with permutation groups is the choice of a data structure for representing coset representatives of the subgroups in the point stabilizer sequence relative to some fixed ordering α of the underlying point set. In this section, we describe a space-efficient data structure, known as a *Schreier vector data structure*, that reduces the worst case time required to access a specified coset representative, together with new algorithms for building this data structure.

Let G be a permutation group acting on an n-element set Ω with G specified by a generating set S, and let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ be a fixed ordering of the points of Ω. The *point stabilizer sequence* of G relative to α is the chain of subgroups

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

where $G^{(i)} = G_{\alpha_1, \ldots, \alpha_{i-1}}$, $1 \leq i \leq n$. S is called a *strong generating set* for G relative to α if

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

The point α_i is called a *base point* relative to α if $|\alpha_i^{G^{(i)}}| \neq 1$. The sequence of points $B = (\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_M})$ consisting of all base points, with $i_1 < i_2 < \ldots < i_M$, is called an *ordered base* for G relative to α. The significance of a base is that each element g of G is uniquely determined by its base image $(\alpha_{i_1}^g, \alpha_{i_2}^g, \ldots, \alpha_{i_1}^g)$. Note that $M \leq \log |G|$. Also, the size of a base may vary with the ordering α, but in general, it is easy to show that two bases relative to two different orderings differ in size by at most a $\log n$ factor.

A *Schreier vector data structure* for G relative to the ordering α is a sequence $\{T_i\}_{i=1}^n$ of *Schreier trees*. Each Schreier tree T_i can be thought of as an ordered pair (R_i, T_i), where T_i is a directed labeled tree, rooted at α_i, with a set of edge labels $R_i \subseteq G^{(i)}$. The nodes of T_i are contained in the orbit $\alpha_i^{\langle R_i \rangle}$. If v is a node of T_i, then the concatenation of the edge labels along the path from α_i to v in T_i is a word in the elements of R_i whose product moves α_i to v. Thus each Schreier tree 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. $\mathcal{T}_i$ is *complete* if the set of nodes of T_i coincides with $|\alpha_i^{G^{(i)}}|$ and the Schreier vector data structure is *complete* if $\mathcal{T}_i$ is complete for $1 \leq i \leq n$. In this case, the Schreier vector data structure defines a *complete transversal system*.

We require the following access functions for a Schreier tree $\mathcal{T}$: $\texttt{Root}(\mathcal{T})$, $\texttt{Labels}(\mathcal{T})$, $\texttt{Nodes}(\mathcal{T})$ and $\texttt{Depth}(\mathcal{T})$. For an arbitrary point v, $\texttt{Coset-rep-as-word}(\mathcal{T}, v)$ returns a word in $\texttt{Labels}(\mathcal{T})$ that moves $\texttt{Root}(\mathcal{T})$ to v if $v \in \texttt{Nodes}(\mathcal{T})$ and returns NIL otherwise. $\texttt{Coset-rep}(\mathcal{T}, v)$ is similar, but returns a permutation rather than a word when possible.

The permutation g is said to *sift through* $\{\mathcal{T}_i\}_{i=1}^n$ if we can write g in the form

$$g = g_{n-1} g_{n-2} \cdots g_1,$$

where g_i is defined recursively as the element $\texttt{Coset-rep}(\mathcal{T}_i, \alpha_i^{g g_1^{-1} \cdots g_{i-1}^{-1}})$. The number of multiplies required to sift an element is proportional to the sum of the depths of the Schreier trees. In the case where the Schreier vector data structure is complete, the above factorization has two important implications. The first is the ability to generate random elements of G according to the uniform distribution. This is achieved by creating an element g whose factorization has the above form with the elements g_i chosen at random according to the uniform distribution from amongst the cosets for $G^{(i+1)}$ in $G^{(i)}$ defined by $\mathcal{T}_i$. The function $\texttt{Random-elt}(\{\mathcal{T}_i\}_{i=1}^n)$ will return a random element constructed in this manner. The second implication is a test for membership in G of an arbitrary permutation. The test consists of attempting to factor an arbitrary permutation through $\{\mathcal{T}_i\}_{i=1}^n$. The factorization will succeed if and only if the element belongs to G. This was Sims's original group membership test [19].

The notion of a Schreier vector data structure was introduced by Sims in order to save up to an order of magnitude of space for typical cases, at the cost of up to an order of magnitude of time in computing coset representatives. 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 $n-1$ and computing a coset representative can take as much as $n-1$ multiplies.

The notion of a monotone Schreier tree was introduced in [3] as a method for reducing the time penalty for accessing coset representatives in Schreier trees without severely increasing the storage requirements. A Schreier tree $\mathcal{T}$ is *monotone* if $\texttt{Labels}(\mathcal{T}) = (g_1, \ldots, g_k)$ is a sequence and the edge labels along the path from $\texttt{Root}(\mathcal{T})$ to each element of $\texttt{Nodes}(\mathcal{T})$ is a word in $(g_1, \ldots, g_k)$ with strictly increasing indices. One consequence of the definition is that $\texttt{Depth}(\mathcal{T}) \leq |\texttt{Labels}(\mathcal{T})|$. Two examples of monotone Schreier trees will first be discussed and then a new construction will be described which is crucial for the results in later sections. The construction of a monotone Schreier tree from a sequence R is a simple modification of a breadth-first search algorithm for building Schreier trees and can be performed in time $O(n|R|)$.

A complete Schreier vector data structure $\{\mathcal{T}_i\}_{i=1}^n$ is said to be *short* if for a fixed constant c, $\sum_{i=1}^n \texttt{Depth}(\mathcal{T}_i) \leq c \log |G|$. It follows from the definition that a short Schreier vector data structure requires the storage of at most $c \log |G|$ permutations of G and $\texttt{Random-elt}(\{\mathcal{T}_i\}_{i=1}^n)$ takes time $O(n \log |G|)$.

Cooperman, Finkelstein and Sarawagi [14] presented a random algorithm to construct a short Schreier vector data structure, with a constant c of 44, from a strong generating set for G. Subsequently the value of c was improved to 21. The key to the result is the following procedure to build short trees, which are monotone. In implementations, one will often use a variation of Build-Short-Schreier-Tree that re-builds the tree in a "breadth-first" manner with the original labels, yielding substantially shorter Schreier trees. The subroutine $\texttt{Extend-Tree}(\mathcal{T}, g)$ extends the monotone Schreier tree $\mathcal{T}$ by appending g to R, applying g to each node of $\mathcal{T}$, and, for each newly discovered node, adding the directed edge labeled by g.

Procedure $\texttt{Build-Short-Schreier-Tree}(\{\mathcal{T}_i\}_{i=1}^n, v)$
Input: A complete Schreier vector data structure $\{\mathcal{T}_i\}_{i=1}^n$ and point v.
Output: A short Schreier tree $\mathcal{T}$ with $\texttt{Root}(\mathcal{T}) = v$.

```
Set T ← (∅, {v}), O ← v^G
While O ≠ Nodes(T) do
    Set g ← Random-elt({T_i}_{i=1}^n)
    Set T ← Extend-Tree(T, g)
Return(T)
```

Proposition 2.1. (from [14, Theorem 3.5]) *In* Build-Short-Schreier-Tree, *let* $\mathcal{O} = v^G$. *Then there exists a constant* $c > 0$ *(we can take* $c = 21$*) such that for all* $\delta \geq 1$, *if* $d = \lceil \delta c \log_2 |\mathcal{O}| \rceil$, Build-Short-Schreier-Tree *will complete after at most* d *random elements have been generated with probability at least* $1 - 1/|\mathcal{O}|^{2\delta}$.

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 $\{g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k} : \epsilon_i \in \{0, 1\}\}$ and $C^{-1}(R) = (C(R))^{-1}$. The cube is *non-degenerate* if $|C(R)| = 2^k$. The idea of a cube was originally introduced by Babai and Szemerédi as part of a doubling trick to build straight-line programs in groups [4]. Their fundamental proposition follows. The proof is easy and is omitted.

Proposition 2.2. *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* $g_{k+1} \notin C^{-1}C$.

A Schreier tree $\mathcal{T}$ is said to be a *cube* Schreier tree if $|\texttt{Labels}(\mathcal{T})|$ and $\texttt{Depth}(\mathcal{T})$ are both bounded by $\log |G|$. Also, $\{\mathcal{T}_i\}_{i=1}^n$ is a *cube* Schreier vector data structure for G if each $\mathcal{T}_i$ is a cube Schreier tree for $G^{(i)}$ for $1 \leq i \leq n$. Using the notion of cubes, it is shown in [3] that given a generating set S for G, it is possible to deterministically build a cube Schreier tree for any orbit of G. The tree constructed was also a monotone Schreier tree. Rather than repeat that argument here, we present a new algorithm which is also based on Proposition 2.2, but which works faster in practice and leads to a better theoretical result.

Proposition 2.3. *Let* S *be a strong generating set for* G. *Then one can compute in time* $O(n \log^2 |G| + n|S| + n \log n)$

both a sequence R of group elements of G such that $C(R)$ is non-degenerate and a complete cube Schreier vector data structure $\{\mathcal{T}_i\}_{i=1}^n$ with the following property. For each $\mathcal{T}_i$, $\texttt{Labels}(\mathcal{T}_i) \subseteq R_i \cup R_i^{-1}$ where $R_i = G^{(i)} \cap R$ is a prefix of R of length at most $\log|G^{(i)}|$, for $1 \leq i \leq n$. In particular, $\{\mathcal{T}_i\}_{i=1}^n$ requires $O(n \log |G|)$ space.

Proof: We may assume that $|S| \leq \log|G|$. Otherwise, this can be accomplished in $O(n|S| + n\log n)$ time by [11, Theorem 2.1]. The code for building the Schreier trees $\mathcal{T}_i$ for $1 \leq i \leq n$ is given below.

```
Build-Cube-Schreier-Vector(S, α)
```
Input: A strong generating set S for G relative to the ordering α
Output: A cube Schreier vector data structure $\mathcal{S}_\alpha$.

```
Initialize R ← ∅
For i ← n downto 1 do
   Set Root(T_i) ← {α_i}, Labels(T_i) ← ∅
   While there exists y ∈ S ∩ G^(i)
         such that Nodes(T_i)^y ≠ Nodes(T_i) do
      Let y ∈ Nodes(T_i) such that y^y ∉ Nodes(T_i)
      Let h ← Coset-rep(T_i, y)
      Append hy to R
      Build a new T_i using breadth-first search
         with R ∪ R^-1 to level 2|R|
Return{T_i}_{i=1}^n
```

We first claim that at any point in its construction, $C(R)$ is a non-degenerate. We prove this by induction on $|R|$. The claim is true when $|R| = 1$ since we never append the identity element to R. For the induction hypothesis, assume that $|R| \geq 1$ and R is non-degenerate. Observe that g' is added to R during the construction of some Schreier tree $\mathcal{T}_i$, because it is discovered that $\alpha_i^{g'} \notin \texttt{Nodes}(\mathcal{T}_i)$. Since $\texttt{Nodes}(\mathcal{T}_i)$ contains all points in $\alpha_i^{C(R^{-1})C(R)}$, it follows that $g' \notin C(R^{-1})C(R)$. Therefore g' doubles the cube $C(R)$ by Proposition 2.2 (i.e $|C(R \cup \{g'\})| = 2|C(R)|$) and so $C(R \cup \{g'\})$ is non-degenerate. This proves the claim. Note also that $|R| \leq \log|G|$.

Since $\{\mathcal{T}_i\}_{i=1}^n$ is constructed in a bottom-up manner, at the time $\mathcal{T}_i$ is built, $R \subseteq G^{(i)}$. By the claim, R is non-degenerate and so $|R| \leq \log|G^{(i)}|$. But $\texttt{Depth}(\mathcal{T}_i) \leq 2|R|$ and $\texttt{Labels}(\mathcal{T}_i) \subseteq R \cup R^{-1}$ by construction and so $\mathcal{T}_i$ is a cube Schreier tree, for $1 \leq i \leq n$.

It remain to verify the timing. We separate the analysis into two phases: a building phase and checking phase. The cost of building any of the trees each time a generator is added is $O(n \log|G|)$ since $|R| \leq \log|G|$ for $1 \leq i \leq n$. This has to be done at most $\log|G|$ times. Thus the total cost of building all the trees is $O(n\log^2|G|)$. The cost of checking if a given tree $\mathcal{T}_i$ is complete is $O(n|S \cap G^{(i)}|) = O(n\log|G|)$ since $|S| \leq \log|G|$. Initially, we must check if α_i is a base point or not. This costs $O(n|S|) = O(n\log|G|)$. Thereafter, we check if the tree is complete only when a new generator has been added and the building phase invoked. This must be done at most $\log|G|$ times and so the checking phase has the same asymptotic time bound as the building phase. This completes the proof. □

3. DETERMINISTIC CYCLIC BASE CHANGE

Let $\alpha = (\alpha_1, \ldots, \ldots, \alpha_n)$ be a fixed ordering and let $\alpha' = (\alpha'_1, \ldots, \alpha'_n)$ be a new ordering obtained from α by a *right cyclic shift*. Thus

$$\alpha' = (\alpha_1, \ldots, \alpha_{r-1}, \alpha_s, \alpha_r, \alpha_{r+1}, \ldots, \alpha_{s-1}, \alpha_{s+1}, \ldots, \alpha_n).$$

A computation that finds a strong generating set for G relative to α' from one for G relative to α is called a (right) *cyclic base change*. (When α' is an arbitrary ordering, the computation is simply called a *base change*.) The following theorem is the main result of this section.

Theorem 3.1. *Let G be a permutation group on n points and let S be a strong generating set for G relative to an ordering α. Let α' be an ordering obtained from α by a right cyclic shift and let $G'^{(i)}$ be the corresponding point stabilizer subgroup. Then one can compute in time $O(n\log^2|G| + n|S| + n\log n)$ a new strong generating set R' relative to α', of size $\log|G|$. Further, R' can be considered as a sequence of group elements of G such that $C(R')$ is non-degenerate, yielding a complete cube Schreier vector data structure $\{\mathcal{T}'_i\}_{i=1}^n$ for G relative to α' with the property that $\texttt{Labels}(\mathcal{T}'_i) \subseteq R'_i \cup R'^{-1}_i$, where $R'_i = G'^{(i)} \cap R'$ is a prefix of R' of length at most $\log|G^{(i)}|$ for $1 \leq i \leq n$. In particular, $\{\mathcal{T}'_i\}_{i=1}^n$ requires $O(n\log|G|)$ space.*

The proof follows the algorithm given in [5] for a cyclic base change but makes use of the Schreier vector data structure. The key to obtaining the stated time and space bounds is the application of Proposition 2.3 for the construction of cube Schreier trees. In the case when $\log|G|$ is substantially smaller than n, this avoids the pathology of creating very deep Schreier trees.

It will be assumed for the remainder of the section that α' is obtained from α by a right cyclic shift as given above. $\Delta^{(i)}$ will denote the i^{th} fundamental orbit $\alpha_i^{G^{(i)}}$ relative to α and $\Delta'^{(i)}$ the i^{th} fundamental orbit $\alpha_i'^{G'^{(i)}}$ relative to α'. Note that for $i < r$ or $i > s$, $\Delta^{(i)} = \Delta'^{(i)}$. For $r < i \leq s$, $\alpha'_i = \alpha_{i-1}$, which implies that $G'^{(i)} = G_{\alpha_1, \ldots, \alpha_{i-2}, \alpha_s}$ and so $\Delta'^{(i)} = \alpha_{i-1}^{G_{\alpha_1, \ldots, \alpha_{i-2}, \alpha_s}} \subseteq \Delta^{(i-1)}$. Also, $\Delta'^{(r)} = \alpha_s^{G^{(r)}}$.

The following result is proved in [5] and helps guide the base change algorithm.

Proposition 3.2. *Let $r < i \leq s$ and let $x = \alpha_{i-1}^{\gamma} \in \Delta^{(i-1)}$ for some $\gamma \in G^{(i-1)}$. Then $x \in \Delta'^{(i)} \iff \alpha_s^{\gamma^{-1}} \in \alpha_s^{G^{(i)}}$.*

Proof of Theorem 3.1:

Proposition 2.3 guarantees us that we can construct a sequence R of group elements of G such that $C(R)$ is non-degenerate and a complete cube Schreier vector data structure $\{\mathcal{T}_i\}_{i=1}^n$ within the stated time bounds. We will build a complete cube Schreier vector data structure $\{\mathcal{T}'_i\}_{i=1}^n$ and a sequence R' satisfying the same property. In order to apply Proposition 3.2, we also require a sequence of cube Schreier trees $\{\mathcal{T}^s_i\}_{i=r+1}^s$ for maintaining the points and coset representatives in the orbit $\alpha_s^{G^{(i)}}$ for $r < i \leq s$. The code for building $\{\mathcal{T}^s_i\}_{i=r+1}^s$ follows

Initialize $\mathcal{T}^s_s \leftarrow \mathcal{T}_s$
Initialize $R^s \leftarrow R \cap G^{(s)}$
For $i \leftarrow s-1$ downto $r+1$ do
  Initialize $\mathcal{T}^s_i \leftarrow$ trivial Schreier tree with root α_s
  While there exists $g \in S \cap G^{(i)}$
      such that $\mathtt{Nodes}(\mathcal{T}^s_i)^g \neq \mathtt{Nodes}(\mathcal{T}^s_i)$ do
    Let $y \in \mathtt{Nodes}(\mathcal{T}^s_i)$ such that $y^g \notin \mathtt{Nodes}(\mathcal{T}^s_i)$
    Let $h = \mathtt{Coset\text{-}Rep}(\mathcal{T}^s_i, y)$
    Append hg to R^s
    Build a new $\mathcal{T}^s_i$ using breadth-first search
      with $R^s \cup R^{s-1}$ to depth at most $2|R^s|$

The proof that the sequence $\{\mathcal{T}^s_i\}_{i=r+1}^s$ satisfies the required properties follows easily once it is shown that the cube for the sequence $C(R^s)$ is non-degenerate. This in turn is proved exactly as in Proposition 2.3 and is omitted.

The construction of $\{\mathcal{T}'_i\}_{i=1}^n$ takes place in three stages, corresponding to the sequences of Schreier trees $\{\mathcal{T}'_i\}_{i=s+1}^n$, $\{\mathcal{T}'_i\}_{i=r+1}^s$ and $\{\mathcal{T}'_i\}_{i=1}^r$ respectively. In the first stage, we simply set $\mathcal{T}'_j = \mathcal{T}_j$ for $s+1 \leq j \leq n$. The most difficult part is the second stage. Here, the observation that $\Delta'^{(i)} \subseteq \Delta^{(i-1)}$, $r < i \leq s$, gives an effective method for determining the i^{th} fundamental orbit $\Delta'^{(i)}$.

Initialize $R' \leftarrow R \cap G^{(s+1)}$
For $i \leftarrow s$ downto $r+1$ do
  Initialize $\mathtt{Root}(\mathcal{T}'_i) \leftarrow \{\alpha'_i\}$, $\mathtt{Labels}(\mathcal{T}'_i) \leftarrow \emptyset$
  [Check if any work needs to be done]
  If $|\Delta^{(i-1)}| > 1$ then
    For each $x \in \mathtt{Nodes}(\mathcal{T}_{i-1})$
        such that $x \notin \mathtt{Nodes}(\mathcal{T}'_i)$ do
      Let $\gamma \leftarrow \mathtt{Coset\text{-}rep\text{-}as\text{-}word}(\mathcal{T}_{i-1}, x)$
      If $\alpha_s^{\gamma^{-1}} \in \mathtt{Nodes}(\mathcal{T}^s_i)$ then
        [Evaluate γ as a permutation]
        Let $\beta \leftarrow \mathtt{Coset\text{-}rep}(\mathcal{T}^s_i, \alpha_s^{\gamma^{-1}})$
        [$\beta\gamma \in G^{(i-1)} \cap G_{\alpha_s} = G'^{(i)}$
          and $\alpha_i'^{\beta\gamma} = \alpha_{i-1}^{\beta\gamma} = x$]
        Append $\beta\gamma$ to R'
        Build a new $\mathcal{T}'_i$ using breadth-first search
          with $R' \cup R'^{-1}$ to depth at most $2|R'|$

The key to the analysis of the second stage is the claim that at any point in its construction, $C(R')$ is non-degenerate. The proof is by induction. The base case is the initial value of R' as $R \cap G^{(s+1)}$ in which case $C(R')$ is non-degenerate by inheritance from R. Let the induction hypothesis be that $C(R')$ is non-degenerate at some intermediate step of the pseudo-code Thereafter $\beta\gamma \in G'^{(i)}$ is added to R' during the construction of some Schreier tree $\mathcal{T}'_i$, because it is discovered that $\alpha_i'^{\beta\gamma} \notin \mathtt{Nodes}(\mathcal{T}'_i)$. Since $\mathtt{Nodes}(\mathcal{T}'_i)$ contains all points in $\alpha_i^{C(R'^{-1})C(R')}$, it follows that $g' \notin C(R'^{-1})C(R')$. Therefore g' doubles the cube $C(R')$ by Proposition 2.2 and so $C(R' \cup \{g'\})$ is non-degenerate. This proves the claim. Note also that $|R'| \leq \log|G'^{(r+1)}|$.

Since $\{\mathcal{T}'_i\}_{i=r+1}^s$ is constructed in a bottom-up manner, at the time $\mathcal{T}'_i$ is built, $R' \subseteq G'^{(i)}$. By the claim, R' is non-degenerate and so $|R'| \leq \log|G'^{(i)}|$. But $\mathtt{Depth}(\mathcal{T}'_i) \leq 2|R'|$ and $\mathtt{Labels}(\mathcal{T}'_i) \subseteq R' \cup R'^{-1}$ by construction and so $\mathcal{T}'_i$ is a cube Schreier tree for $r < i \leq s$.

We now analyze how long the second stage takes. Let b' be the number of fundamental orbits $\Delta^{(i-1)}$ which are non-trivial for $r < i \leq s$. Then $O(nb')$ points have to be examined to test for inclusion in fundamental orbits relative to α. The cost of each test is $O(\log|G|)$. This involves the evaluation of the word γ^{-1} on a single point. Since the word has length $O(\log|G|)$, the total cost of all tests is $O(nb'\log|G|) = O(n\log^2|G|)$.

When a new element is added, this requires the multiplication of a word of length $O(\log|G|)$. The total number of new elements that are added cannot exceed $|R'| \leq \log_2|G'^{(r+1)}| = O(\log|G|)$. Thus the total cost for adding new elements in this portion of the code is $O(n\log^2|G|)$.

The Schreier tree $\mathcal{T}'_i$ needs to be updated only when a new generator is added. Since this occurs at most $\log|G|$ times, the total cost of building the trees is $O(n\log^2|G|)$ time. Thus, the second stage operates within the required time and space bounds.

We complete the proof by indicating how to complete the third stage during which the remaining trees $\{\mathcal{T}'_i\}_{i=1}^r$ are built. This is easy to do in light of previous developments because $G'^{(i)} = G^{(i)}$ for $1 \leq i \leq r$. The code to do this follows, and is almost identical to `Build-Cube-Schreier-Vector` of Proposition 2.3. At this point, R' will have the value assigned to it at the end of the second stage.

For $i \leftarrow r$ downto 1 do
  Initialize $\mathtt{Root}(\mathcal{T}'_i) \leftarrow \{\alpha'_i\}$, $\mathtt{Labels}(\mathcal{T}'_i) \leftarrow \emptyset$
  While there exists $g \in S \cap G^{(i)}$
      such that $\mathtt{Nodes}(\mathcal{T}'_i)^g \neq \mathtt{Nodes}(\mathcal{T}'_i)$ do
    Let $x \in \mathtt{Nodes}(\mathcal{T}'_i)$ such that $x^g \notin \mathtt{Nodes}(\mathcal{T}'_i)$
    Let $h \leftarrow \mathtt{Coset\text{-}Rep}(\mathcal{T}'_i, x)$
    Append hg to R
    Build a new $\mathcal{T}'_i$ using breadth-first search
      with $R \cup R^{-1}$ to depth at most $2|R|$

The argument that $C(R')$ is a non-degenerate and the resulting implication that the Schreier trees $\mathcal{T}'_i$, $1 \leq i \leq r$ are cube Schreier trees is almost identical to the argument given for the proof of Proposition 2.3 and is omitted. For the same reason, we omit the proof that the time and space requirements are $O(n\log^2|G|)$ and $O(n\log|G|)$ respectively. We now conclude that R' and $\{\mathcal{T}'_i\}_{i=1}^n$ satisfy the conclusion of Theorem 3.1 and that the algorithm works within the stated time and space requirements. □

An interesting corollary to Theorem 3.1 is a fast method for determining the number of nodes in the fundamental orbits for an ordering α' obtained from α by a right cyclic shift when a complete short Schreier vector data structure is provided. This will be useful in section 4 where a fast randomized cyclic base change algorithm will be presented.

Corollary 3.3. *Let $\{\mathcal{T}_i\}_{i=1}^n$ be a complete short Schreier vector data structure for G relative to an ordering α. Let α' be an ordering obtained from α by a right cyclic shift Then $n'_i = |\Delta'^{(j)}|$, $1 \leq i \leq n$ can be determined in time $O(n\log|G|)$.*

Proof: (Sketch) Assume that α and α' are given as before. It suffices to determine the values of $|\Delta'^{(j)}|$ for $r \leq j \leq s$.

Note that $\Delta'^{(r)} = \alpha_s^{G^{(r)}}$, which can be computed within the given time bound. The key to the remaining fundamental orbits is Proposition 3.2. In order to determine $\Delta'^{(j)}$ as j goes from s down to $r+1$, note that each $x \in \Delta^{(j-1)}$ defines a $\gamma = \texttt{Coset-rep-as-word}(\mathcal{T}_{j-1}, x)$ and we must check whether $\alpha_s^{\gamma^{-1}} \in \alpha_s^{G^{(j)}}$. By hypothesis, γ is a word in $\texttt{Labels}(\mathcal{T}_{j-1})$ of length at most $\texttt{Depth}(\mathcal{T}_{j-1})$. There are at most n such tests for each $\Delta^{(j-1)}$. Each evaluation of $\alpha_s^{\gamma^{-1}}$ requires $O(\texttt{Depth}(\mathcal{T}_j))$ time and $\sum_{i=1}^{n} \texttt{Depth}(\mathcal{T}_j) = O(\log |G|)$. Thus the total cost of all such evaluations is $O(n \log |G|)$. Testing if $\alpha_s^{\gamma^{-1}} \in \alpha_s^{G^{(j)}}$ takes constant time once the orbits $\alpha_s^{G^{(j)}}$ have been constructed. This can be done in a bottom-up manner using an incremental breadth-first search with the computation organized so that each generator of $G^{(r)}$ is never applied more than once to any point of $\Delta^{(r)}$. (See [5] for a more detailed description.) Since there are at most $c \log |G^{(r)}|$ generators for $G^{(r)}$, which appear as edge labels of $\mathcal{T}_j$, $r \leq j \leq s$, it takes $O(n \log |G|)$ time to complete this phase. Thus the total running time is as stated. □

4. RANDOMIZED CYCLIC BASE CHANGE

The new deterministic cyclic base change has the same time and space complexity as the general randomized base change [14]. This leads to the natural question, whether there is a faster randomized cyclic base change. The next theorem answers the question in the affirmative.

Theorem 4.1. *Let G be a permutation group acting on n points and let $\{\mathcal{T}_i\}_{i=1}^{n}$ be a complete short Schreier vector data structure for G relative to an ordering α. Let α' be an ordering obtained from α by a right cyclic shift. Then one can build a complete short Schreier vector data structure $\{\mathcal{T}'_i\}_{i=1}^{n}$ for G relative to α'. With probability at least $1 - 2/n$, the algorithm completes in time $O(nb \log^2 n)$, for b the size of a non-redundant base with respect to α or α'. Further, each Schreier tree $\mathcal{T}'_i$ will be of depth at most $6.3 \log n_i$, for n_i the size of the i^{th} fundamental orbit.*

The proof of this theorem is deferred, while a crucial lemma and the necessary procedures are described.

Lemma 4.2. *Let H be a subgroup of a finite permutation group G and let U be a complete set of right coset representatives for H in G. For $g \in H$, let $\bar{g}$ be the unique element of U so that $H\bar{g} = Hg$. Let g be a uniformly random element of G. The the element $g\bar{g}^{-1}$ is uniformly random in H. Furthermore, if $S \subseteq G$ is a set of mutually independent uniformly random elements of G, then $T = \{g\bar{g}^{-1} : g \in S\}$ is a set of mutually independent and uniformly random elements of H.*

Proof: To prove the first part, assume that g is uniformly random in G. Given an arbitrary $h \in H$, there are exactly $|U|$ elements of G which will produce h, namely the elements of the set hU. Since $\text{Prob}(g \in hU) = |U|/|G|$, it follows that $\text{Prob}(g\bar{g}^{-1} = h) = 1/|H|$. To prove the second part, let $S = \{g_1, \ldots, g_k\}$. Then

$$\begin{aligned}&\text{Prob}((g_1\bar{g}_1^{-1} = h_1) \wedge \cdots \wedge (g_k\bar{g}_k^{-1} = h_k))\\ =&\text{Prob}((g_1 \in h_1U) \wedge \cdot \wedge (g_k \in h_kU)) = 1/|H|^k\end{aligned}$$

since $g_1, \cdots, g_k$ are mutually independent by assumption. Thus $g_1\bar{g}_1^{-1}, \ldots, g_k\bar{g}_k^{-1}$ are independent and the second part is proved. □

The previous lemma is important for understanding the correctness of the randomized cyclic base change algorithm, below. For Schreier trees for G acting on Ω, with $\mathcal{T}$ and $\mathcal{T}_i$ Schreier trees, $\alpha \in \Omega$, and $g \in G$, the routines $\texttt{Labels}(\mathcal{T})$, $\texttt{Depth}(\mathcal{T})$, $\texttt{Coset-rep}(\mathcal{T}, \alpha)$, and $\texttt{Random-elt}(\{\mathcal{T}_i\}_{i=r}^{n})$ were defined in section 2. Their implementation is clear from the definitions. The time required for `Coset-rep` is bounded by the depth of $\mathcal{T}$ times the time for a permutation multiplication. We additionally define the routine $\texttt{Size}(\mathcal{T})$, which returns the number of $\alpha \in \Omega$ for which `Coset-rep` returns a non-`NIL` value.

Finally, we require, for technical reasons, a modified routine $\texttt{Extend-Tree-if-Success}(\mathcal{T}, g, n)$, where n is a number. In applications, $\mathcal{T}$ will be a Schreier tree for an orbit of $G^{(i)}$ for some i, and n will correspond to the orbit length. The routine `Extend-Tree-if-Success` should call $\texttt{Extend-Tree}(\mathcal{T}, g)$ as defined in section 2, but only if g is a *success*, as defined below.

Definition. (from [14, Theorem 3.5]) *Let P be the nodes of a Schreier tree $\mathcal{T}$ for an orbit of length n. The group element g is a success for $\mathcal{T}$ if either*
$|P| \leq n/2$ and $|P^g - P| \geq |P|/4$ or
$|P| \geq n/2$ and $|P^g - P| \geq (n - |P|)/4$

Note that for an orbit of length n_i and an initial trivial Schreier tree, and then modifying it by adding $O(\log n_i)$ successes relative to the current Schreier tree suffices to build a Schreier tree with all n_i nodes. This motivates the following proposition.

Proposition 4.3. *For G acting on a set of size n and for some i with i with $1 \leq i \leq n$, if one knows the index $n_i = [G^{(i)} : G^{(i+1)}]$ and has $r \geq 21 \log n$ mutually independent, random elements of G (but not necessarily generating G), then one can form a short Schreier tree of depth at most $6.3 \log_2(n_i/2)$ for $G^{(i)}/G^{(i+1)}$ in time $O(nr)$ with probability at least $1 - 1/n^2$.*

Proof: Let $\mathcal{O}$ be the orbit of α_i under $G^{(i)}$. By Proposition 2.1 for $\delta = \log_2 n / \log_2 |\mathcal{O}|$, one can construct a Schreier tree of depth $d = 21\delta \log_2 |\mathcal{O}| = 21 \log_2 n$ with probability at least $1 - 1/n^2$. The depth is too large for our purposes. However, inspection of the original proof of [14, Theorem 3.5], shows that one can choose a subset of the r random elements (the "successes") of size at most $2\log_{5/4}(n_i/2) = 2\log_2(n_i/2)/\log_2(5/4) < 6.3 \log_2(n_i/2)$, which will also build the Schreier tree with the stated probability. □

The next procedure contains the key idea of this section. It is called to incrementally construct short Schreier trees in two distinct cases: for the orbits $\alpha_s^{G^{(i)}}$ for $r \leq i \leq s$ in the procedure `Random-Cyclic-Base-Change`, and for the fundamental orbits $\alpha_i^{G^{(i)}}$ for $1 \leq i \leq n$ in the proof of Corollary 4.4. As we shall see, `Augment-Trees` is more efficient than an analogous routine described in [12, 14], since all of the Schreier trees under construction can be augmented

within the asymptotic time needed to construct a single random group element.

Augment-Trees($\{\overline{T}_i\}_{i\in B}, \{\overline{n}_i\}_{i\in B}, B, \{T_i\}_{i=r}^n$)
Input: A collection of (possibly incomplete) Schreier trees $\{\overline{T}_i\}$ indexed by a subset B, with each tree representing the orbit $(\mathrm{Root}(\overline{T}_i))^{G^{(i)}}$, the *known* sizes $\overline{n}_i = \left|(\mathrm{Root}(\overline{T}_i))^{G^{(i)}}\right|$, and a complete Schreier vector data structure $\{T_i\}_{i=r}^n$ such that $B \subseteq [r, n]$.
Output: No output, but $\{\overline{T}_i\}_{i\in B}$ is modified

Initialize $g \leftarrow$ Random-elt($\{T_i\}_{i=r}^n$)
For $j \leftarrow r$ to n
  If $\alpha_j \in B$
    Extend-Tree-if-Success($\overline{T}_j, g, \overline{n}_j$)
    Set $h \leftarrow$ Coset-rep($T_j, {\alpha_j}^g$)
    Set $g \leftarrow gh^{-1}$

An interesting application of Augment-Trees is to construct a complete Schreier vector data structure in which each Schreier tree is short.

Corollary 4.4. *Given a short Schreier vector data structure $\{T_i\}$ with $\sum_{i=1}^n \mathrm{Depth}(T_i) \le c \log |G|$, one can form short Schreier trees of depth at most $6.3 \log n_i$ for all $1 \le i \le n$ (where $n_i = \mathrm{Size}(T_i)$) in time $O(nc \log |G| \log n)$ with probability at least $1 - 1/n$.*

Proof: Let $\{\overline{T}_i\}_{i=1}^n$ be the short Schreier trees that one wishes to construct. The sizes of the fundamental orbits $n_i = |\alpha_i^{(\mathrm{Labels}(\overline{T}_i))}|$ are the same as for T_i and so are known in advance. Initialize $\overline{T}_i$ to the trivial tree for each i, and make $21 \log n$ calls to Augment-Trees($\{\overline{T}_i\}_{i=1}^n, \{n_i\}_{i=1}^n, \Omega, \{T_i\}_{i=1}^n$). By repeated application of Lemma 4.2, the set of g at level j computed by the algorithm over all $21 \log n$ calls is mutually independent. By Proposition 4.3, each $\overline{T}_i$ has depth at most $6.3 \log_2(n_i/2) \le 6.3 \log_2 n$ with probability at least $1 - 1/n^2$. Random elements at different levels can be correlated. (Consider, for example, what would happen to a random element equal to the identity.) Nevertheless, the probability of an error at any level is at most n times the probability of error on a particular level. So, with probability at least $1 - 1/n$, all Schreier trees will have the indicated depth. □

Finally, the randomized cyclic base change can be presented. As in section 3, we assume that α' is a new ordering obtained from α by a right cyclic shift of the subsequence $\{\alpha_r, \ldots, \alpha_{s-1}, \alpha_s\}$ to $\{\alpha_s, \alpha_r, \ldots, \alpha_{s-1}\}$, and that $G'(i)$ is determined with respect to the ordering α'.

Random-Cyclic-Base-Change($\{T_i\}_{i=1}^n, r, s$)
Input: complete Schreier trees T_i for $G^{(i)}/G^{(i+1)}$ such that $\sum_{i=1}^n \mathrm{Depth}(T_i) \le c \log |G|$ for some c, and indices r, s satisfying $1 \le r \le s \le n$
Output: complete Schreier trees T'_i for $G'^{(i)}/G'^{(i+1)}$ of depth at most $6.3 \log n'_i$ (where the prime denotes with respect to the cyclically permuted ordering α' of the points of Ω, and $n'_i = [G'^{(i)} : G'^{(i+1)}]$)

For $j \leftarrow 1$ to n
  Set $n'_j = |\Delta'^{(j)}|$ $(\equiv |{\alpha'_j}^{G'^{(j)}}|)$ [via Corollary 3.3]
  Initialize $T'_j \leftarrow$ trivial Schreier tree with root α'_j
For $j \leftarrow r$ to s
  Set $n^s{}_j = |\alpha_s^{G^{(j)}}|$ [in $O(n \log |G|)$ time]
  Initialize $T^s{}_j \leftarrow$ trivial Schreier tree with root α_s
Initialize $B \leftarrow \{\alpha_s\} \cup$ base points of $\{T_i\}_{i=r}^s$
While Size(T'_i) $< n'_i$ for some i do
  Set $g \leftarrow$ Random-elt($\{T_i\}_{i=1}^n$)
  For $j \leftarrow 1$ to n
    If $n'_j > 0$
      If $j < r$ or $j > s$
        Extend-Tree-if-Success(T'_j, g, n'_j)
      Else if $r < j \le s$
        [For efficiency, compute $\{T^s{}_i\}$ only as necessary (by lazy evaluation) instead of pre-computing all of $\{T^s{}_i\}$]
        While Coset-rep($T^s{}_{j-1}, {\alpha_s}^g$) = NIL
          Augment-Trees($\{T^s{}_i\}_{i\in B}, \{n^s{}_i\}_{i\in B}, B, \{T_i\}_{i=r}^n$)
        Set $h \leftarrow$ Coset-rep($T^s{}_{j-1}, {\alpha_s}^g$)
        Extend-Tree-if-Success(T'_j, gh^{-1}, n'_j)
      Else [when $j = r$]
        Augment-Trees($\{T^s{}_r\}, \{n^s{}_r\}, \{\alpha_r\}, \{T_i\}_{i=r}^n$)
        Set $T'_r \leftarrow T^s{}_r$
      Set $h \leftarrow$ Coset-rep($T_j, {\alpha_j}^g$)
      Set $g \leftarrow gh^{-1}$
Return $\{T'_i\}_{i=1}^n$

Proof of Theorem 4.1:
We will show that $O(\log n)$ iterations of the outermost "while" loop suffice for completion of the algorithm. In addition, there will be at most $O(\log n)$ calls to Augment-Trees over the life of the algorithm. The probability of failing to construct a short Schreier tree T'_j of depth at most $6.3 \log n'_j \le 6.3 \log n$ can be shown to be at most $1/n^2$ in the same manner as in the proof of Proposition 4.3. A similar case holds for $T^s{}_j$, where $n^s{}_j \le n$. So, a proof similar to Corollary 4.4 for $\{T^s{}_i\}$ shows that with probability at least $1 - 2/n$, one can construct Schreier trees T'_j of depth at most $6.3 \log n'_j$ and Schreier trees $T^s{}_j$ of depth at most $6.3 \log n$ with overall probability at least $1 - 2/n$. (Note that one cannot replace $\log n$ by $\log n_j$ for the depth of $T^s{}_j$, and there are examples in which the sum of the depths of $T^s{}_j$ for all j may exceed $O(\log |G|)$.)

It remains to verify the time. Computing $\{n'_i\}_{i=1}^n$ requires $O(nb \log n)$ time by Corollary 3.3. Computing $\{n^s{}_i\}_{i=r}^s$ can be done via the bottom-up construction of $\alpha_s^{G^{(j)}}$ in the proof of Corollary 3.3 and requires $O(n \log |G|)$ time. As already shown, with the stated probability only $O(\log n)$ random elements g will be constructed. The cost of creating each random element and multiplying by coset representatives Coset-rep($T_j, {\alpha_j}^g$) will be $O(n \log |G|)$, since $\{T_i\}_{i=1}^n$ forms a short Schreier vector data structure. The cost of multiplying by coset representatives Coset-rep($T^s{}_j, {\alpha_s}^g$) is $O(nb \log n)$, where b is the base size, or the number of non-trivial trees $\{T_i\}$. Thus, there are $O(\log n)$ random elements, and the time related to each one is $O(nb \log n)$, yielding the overall time. □

Note that in common with most base change algorithms, the new cyclic randomized algorithm is Las Vegas, in the sense that one can deterministically verify if the answer is correct. The verification is done by comparing the group orders as computed by the original and new strong generating sets. We have previously stated a general random-

ized base change algorithm [14] with asymptotic complexity $O(n \log^2 |G|)$, that is also Las Vegas.

5. FAST CONSTRUCTION OF SHORT SCHREIER TREES FROM A STRONG GENERATING SET

Both the result of the previous section and the original randomized cyclic base change [14] required as input short Schreier vector data structures (satisfying $\sum_{1 \le i \le n} \text{depth}(\mathcal{T}_i) = O(\log |G|)$), in order to guarantee that the output Schreier vectors have depth $O(\log n_i)$. This requirement can be removed by Theorem 5.2 below, which provides a recipe for quickly constructing short Schreier vector data structures. In fact, a stronger result is achieved in constructing short Schreier trees (depth $O(\log n_i)$) from the strong generating set alone. (Recall that $n_i = [G^{(i)} : G^{(i+1)}]$.)

The following result on reduction of a strong generating set is well-known, and is included for completeness. In most applications, $O(\log n) \ll O(\log |G|)$. In such situations, [11, Theorem 2.5] shows how to find a reduced strong generating set in time $O(n|S| + n \log n)$, and will usually be faster in implementations.

Lemma 5.1. *Given a strong generating set S for a group finite G, one can construct a new strong generating set $S' \subseteq S$ with $|S'| \le \log |G|$ in $O(n|S| + nb|S'|)$, where b is the size of the smallest non-redundant base with respect to the ordering of S.*

Proof: The following pseudo-code fragment has the correct properties.

```
Initialize S' ← ∅
For i ← n − 1 downto 1
    For g ∈ S ∩ G^(i) − G^(i+1) do
        If α_i^({g}∪S') ≠ α_i^(S')
            Set S' ← S' ∪ {g}
Return(S')
```

In implementations, for each level i one computes the set $\alpha_i^{\{g\} \cup S'}$ incrementally from $\alpha_i^{\langle S' \rangle}$. Further, for each i, the elements of $S \cup S'$ should be applied to each point at most once. Each element of S will be invoked for at most one level, i. From this, the time follows. After adding each g, the size of $\langle S' \rangle$ must at least double. Hence, $|S'| \le \log |G|$. □

Theorem 5.2. *For a permutation group G of degree n with strong generating set S, one can construct all short Schreier trees for $G^{(i)}/G^{(i+1)}$ for $1 \le i \le n$ in time $O(n \log^2 |G| + n|S|)$ of depth at most $6.3 \log n_i$, with probability at least $1 - 1/|G|$. Further, these short Schreier trees form a group membership data structure from which random elements can be derived in time $O(n \log |G|)$.*

Proof: The conclusion on the time to build random elements follows from noting that a random group element is constructed from the product of at most $n-1$ random coset representatives in time $\sum_{1 \le i \le n-1} O(n \log n_i) = O(n \log |G|)$. It suffices to prove the time and probability for construction of short Schreier trees when $|S| \le \log |G|$ by Lemma 5.1.

By induction on j, we assume monotonic short Schreier trees for $G^{(i)}/G^{(i+1)}$ with $j < i \le n$ have been built, and we will construct such a tree for $G^{(j)}/G^{(j+1)}$. A (trivial) short Schreier tree for $G^{(n)}/G^{(n+1)}$ can be built in $O(1)$ time. Let $T^{(i)}$ for $j < i \le n$ be the set of coset representatives for $G^{(i)}/G^{(i+1)}$, and let $D^{(j+1)} = T^{(n)} T^{(n-1)} \cdots T^{(j+1)} = G^{(j+1)}$. Note that each element of $D^{(j+1)}$ can be built as a word of length $O(\log |G^{(j+1)}|)$, since each $T^{(i)}$ is built from a Schreier tree with depth $O(\log n_i)$.

One first pre-computes, in time $O(n \log |G^{(j)}| \log n_j)$, a data structure which allows computation of a coset representative of $G^{(j)}/G^{(j+1)}$ in time $O(n \log |G^{(j)}|)$. (This data structure will then enable construction of a short Schreier tree for $G^{(j)}/G^{(j+1)}$.) Let U be initialized to the identity. By Proposition 2.2,

$$|D^{(j+1)} U \{g, 1\}| = 2|D^{(j+1)} U|$$
$$\iff g \notin U^{-1} (D^{(j+1)})^{-1} D^{(j+1)} U = U^{-1} G^{(j+1)} U.$$

If there is a $s \in S \cap G^{(j)}$ such that $\alpha_j^{U^{-1} G^{(j+1)} U s} \ne \alpha_j^{U^{-1} G^{(j+1)} U}$, then construct a $g \in U^{-1} G^{(j+1)} U s - U^{-1} G^{(j+1)} U$. Replace U by $U\{g, 1\}$. Repeat until $\alpha_j^{U^{-1} G^{(j+1)} U s} = \alpha_j^{U^{-1} G^{(j+1)} U}$ for all $s \in S \cap G^{(j)}$. This can be repeated at most $\log n_j$ times since $\log |G^{(j+1)}| \le |D^{(j+1)} U| \le \log |G^{(j)}|$, and $|D^{(j+1)} U|$ is doubled each time. So, the length of any word in $U^{-1} D^{(j+1)} U$ is at most $O(\log |G^{(j)}|)$. Thus, g and any coset representative for $G^{(j)}/G^{(j+1)}$ can be computed in time $O(n \log |G|)$ (by retaining pointer data structures from previous levels).

Given the availability of coset representatives for $G^{(j)}/G^{(j+1)}$ in time $O(n \log |G|)$ and the availability of short Schreier trees (depth $O(\log n_i)$) for $G^{(i)}/G^{(i+1)}$ for $j < i \le n$, one can construct random elements of $G^{(j)}$ in time $O(n \log |G^{(j)}|)$. $6.3 \log n_j$ such random elements that are successes (in the sense of section 4) can be used to build a new Schreier tree $\mathcal{T}_j$, for $G^{(j)}/G^{(j+1)}$, which is short. Thus, the time is $O(n \log |G| \log n_j)$ at level j, or $O(n \log^2 |G|)$ overall. This completes the induction on j.

It remains to verify the reliability. $6.3 \log_2 |G|$ successes among all levels will suffice to build short Schreier trees at all levels. With probability at least $p = 1/3$, the random element will be a success (in the sense of Proposition 4.3 and the previous definition) with respect to $\mathcal{T}_j$. Since the fundamental orbit lengths n_i are known in advance, one can discard those random elements that are not successes. Chernoff's Bound [9] states that for S_t the number of successes in t independent Bernoulli trials with probability of success p, and for $0 < \epsilon < 1$,

$$Prob(S_t \le \lfloor (1 - \epsilon) pt \rfloor) \le e^{-\epsilon^2 pt/2}$$

Choosing $\epsilon = 1/2$, $t = 54.5 \ln |G| = 37.8 \log_2 |G|$ and noting $p = 1/3$ then yields $(1 - \epsilon) pt = (37.8/6) \log_2 |G| = 6.3 \log_2 |G|$, the required number of successful trials. Chernoff's bound predicts a reliability of $e^{-\epsilon^2 pt/2} \ge 1 - 1/|G|^2 .27 \ge 1 - 1/|G|$.

On completion of the algorithm, $D^{(1)}$ will be of length at most $6.3 \log |G|$, requiring $O(n \log^2 |G|)$ time for

its construction. The n short Schreier trees also require $O(n \log^2 |G|)$ time. □

The proof used $37.8 \log_2 |G|$ iterations to guarantee the $6.3 \log_2 |G|$ successes with the indicated worst-case reliability. In implementations using breadth-first search, many fewer than $6.3 \log_2 |G|$ successes will suffice, and a larger fraction of the trials will be successes than would be indicated by the worst case analysis. The previous result allows us to strengthen our arbitrary randomized base change [12, 14; Theorem B].

Theorem 5.3. *Given a strong generating set S of size $O(\log |G|)$ for a group G, there is a randomized base change algorithm that, with probability $1 - 1/|G|$, constructs a short Schreier vector data structure in $O(n \log^2 |G| + n|S| \log |G|)$ time.*

Proof: Lemma 5.1 is first applied to find a strong generating set of size at most $\log |G|$, followed by Theorem 5.2 to generate random group elements in $O(n \log |G|)$. The hypothesis of [14, Theorem B] is then satisfied, yielding the conclusion of that theorem. □

REFERENCES

1. L. Babai, "Monte-Carlo Algorithms in Graph Isomorphism Testing", Université de Montréal Tech. Report D.M.S. 79-10 (1979), Dep. Math. et Stat.
2. L. Babai, G. Cooperman, L. Finkelstein, E.M. Luks, and Á. Seress, "Fast Monte Carlo Algorithms for Permutation Groups", *Proc. 23^{rd} ACM STOC* (1991), pp. 90–100.
3. L. Babai, G. Cooperman, L. Finkelstein, and Á. Seress, "Nearly Linear Time Algorithms for Permutation Groups with a Small Base", *Proc. of the 1991 International Symposium on Symbolic and Algebraic Computation* (ISSAC '91), Bonn, pp 200–209, July, 1991.
4. L. Babai and E. Szemerédi, "On the Complexity of Matrix Group Problems I," *Proc. 25^{th} IEEE FOCS* (1984), Palm Beach, FL, pp. 229–240.
5. 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.
6. C.A. Brown, L. Finkelstein, and P.W. Purdom, "Backtrack Searching in the Presence of Symmetry", *Proc. of the 6^{th} International Conference on Algebraic Algorithms and Error Correcting Codes* (AAECC-6, Rome, 1988), Springer Verlag Lecture Notes in Computer Science, Vol. 357, pp 99–110
7. G. Butler and C. Lam, "Isomorphism Testing of Combinatorial Objects", *J. of Symbolic Computation* **1**, (1985), pp 363–381.
8. J.J Cannon, "An Introduction to the Group Theory Language, Cayley", in *Computational Group Theory*, edited by M.D. Atkinson, Academic Press, 1984, pp. 145–184
9. H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations", *Annals of Math Statistics* **23**, 1952, pp 493–507.
10. G. Cooperman and L. Finkelstein, "New Methods for Using Cayley Graphs in Interconnection Networks", *Discrete Applied Mathematics*, Special Issue on Interconnection Networks, in press.
11. G. Cooperman and L. Finkelstein, "A Strong Generating Test and Short Presentations for Permutation Groups", *J. Symbolic Computation* **12** (1991), pp. 475–497.
12. G. Cooperman and L. Finkelstein, "A Random Base Change Algorithm for Permutation Groups", *J. Symbolic Computation*, under revision.
13. G. Cooperman, L. Finkelstein and E. Luks, "Reduction of Group Constructions to Point Stabilizers", *Proceedings of the International Symposium on Symbolic and Algebraic Computation* (ISSAC 89), ACM Press, pp. 351–356.
14. G. Cooperman, L. Finkelstein and N. Sarawagi, "A Random Base Change Algorithm for Permutation Groups", *Proc. of 1990 International Symposium on Symbolic and Algebraic Computation*, (ISSAC 90, Tokyo), ACM Press and Addison-Wesley (1990), pp. 161–168.
15. M. Jerrum, "A Compact Representation for Permutation Groups", *J. Algorithms* **7** (1986), pp. 60–78.
16. C.W.H. Lam, "The Search for a Finite Projective Plane of Order 10", American Mathematical Monthly **98**, (1991), pp. 305–318.
17. 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), pp. 941–974.
18. J. Leon, "Computing Automorphism Groups of Combinatorial Objects", in *Computational Group Theory*, edited by M. D. Atkinson, Academic Press (1984), pp. 321–337.
19. C.C. Sims, "Computation with Permutation Groups", in *Proc. Second Symposium on Symbolic and Algebraic Manipulation*, edited by S.R. Petrick, ACM Press, New York, 1971, pp. 23–28.