Random Algorithms for Permutation Groups

Gene Cooperman* and Larry Finkelstein*
College of Computer Science
Northeastern University
360 Huntington Ave.
Boston, Mass. 02115
e-mail: *gene@corwin.ccs.northeastern.edu*
laf@dworkin.ccs.northeastern.edu

A variety of new combinatorial techniques for computing with groups is reviewed. It is shown how to apply these techniques to construct randomized algorithms for solving fundamental problems. These are group membership and base change (for permutation groups) and normal closure (for both permutation groups and matrix groups over finite fields). As with any randomized algorithm, the reliability must be characterized. This is done in terms of the concepts of Monte Carlo and Las Vegas defined in the text.

1. Introduction

Randomization has long been recognized as a powerful tool that can significantly speed up algorithmic solutions to important problems in computational group theory. This has been a strong motivation for our own work. Unfortunately, many previous attempts to use randomness are heuristic, and lack theoretical results on the running time or on the probability of correctness. A theorem concerning correctness puts the algorithm on a sounder mathematical basis. This often provides the basis for improving the algorithm based on a greater knowledge of internal mathematical structure, rather than having to rely on more empirical techniques.

While the desirability of theoretical guarantees of correctness is recognized, such theorems typically elude researchers for one of three reasons:

- The distribution of the chosen "random" elements may not be characterized by a probabilistic distribution suitable for analysis;

*The work of these authors was supported in part by the National Science Foundation under grant number CCR-8903952.

- an a priori bound may not be known on the probability that a random element from a known distribution satisfies a required property; or
- a bound (probabilistic or deterministic) on the required number of operations cannot be provided.

There are several important examples in the literature, which currently rely on such heuristic techniques. A well-known technique dating back to Jordan ([34], Theorem 13.9) recognizes whether a subgroup G of S_n, specified by a set of generators, contains A_n. The test consists of first verifying primitivity, which is fast, and then attempting to find an element with a special cycle structure. A similar approach applies to a recent innovative method of Neumann and Praeger [31] for recognizing when a subgroup of the general linear group $GL(n,q)$ specified by a set of generating elements "essentially" contains $SL(n,q)$. Along similar lines is the well-known heuristic for finding an element of prime order p in a permutation group G, where p divides $|G|$. The algorithm selects randomly chosen elements until a p-singular element is found. Only recently was this heuristic justified by a result of Kantor, Seitz and Spaltenstein, showing that the probability of success is at least $1/n$. Finally, Leon's random Schreier-Sims algorithm [29] is extremely fast and accurate in practice, but it relies on purely empirical parameters, and never provides a theoretical estimate of the probability of correctness of its answer.

We say that a Monte Carlo algorithm is a randomized algorithm whose reliability (probability of success) can be increased at the cost of additional time. A Monte Carlo algorithm is Las Vegas, if it never returns an incorrect answer. (Hence, a Las Vegas algorithm may only return a correct answer or "don't know".) An algorithm is exponentially reliable if the probability of returning an incorrect answer or "don't know" is bounded above by $\exp(-n)$, for n the input parameter.

We now present an overview of recent developments in Monte Carlo algorithms for computing with groups. Some of the key ideas described have been developed in collaboration with our colleagues Láslo Babai, Eugene Luks and Ákos Seress.

In this paper, we focus on algorithms for solving three fundamental problems involving permutation groups of an n-element set Ω.

- Compute a strong generating set relative to a fixed ordering of Ω for a subgroup $G = \langle S \rangle$ of $\mathrm{Sym}(\Omega)$. This is analogous to putting a matrix in upper triangular form.

- Given a strong generating set S for $G \subseteq \mathrm{Sym}(\Omega)$ relative to an ordering α, compute a strong generating set S' for G relative to a new ordering α'. This is a *change of base*, analogous to a change of basis for a vector space.

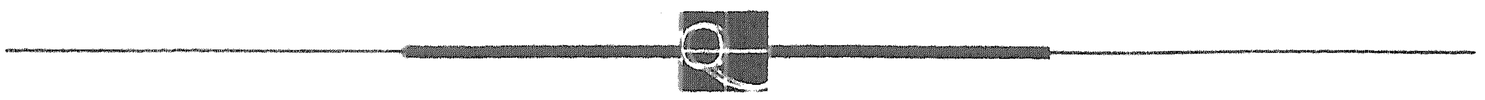
- Given subgroups $H = \langle S \rangle$ and $G = \langle T \rangle$ of $\mathrm{Sym}(\Omega)$, find a generating set for the normal closure $\langle H^G \rangle$ of H under G. (Group operations on G mod H (G/H) are well-defined only when H is normal in G. The normal closure of H under G is the smallest normal subgroup of $\langle H, G \rangle$ containing H)

These problems are fundamental to computational group theory and arise naturally in the course of attempting to solve higher level problems. Kantor's article in this issue [28] provides several excellent examples of their use. Our approach will show that these problems are very much related and that progress on one front usually results in progress on a different front. For example, fast Monte Carlo algorithms for performing base change [20] and normal closure [5] were key ingredients in achieving improved running times for Monte Carlo group algorithms for testing membership in both "small" and "large" base permutation groups.

In Section 2, we present a general framework for the incremental construction of subgroups through random elements. It is well known how to do this with uniformly distributed random elements ([2], [29]), but the construction of such elements appears no easier than testing membership. What is significant and new in the methods described in this paper is that for certain fundamental problems, it is not necessary to have uniformly distributed random elements. We will show that group elements drawn from a very different distribution, called *random subproducts*, can be used to simulate those properties of uniformly distributed random elements required for such constructions but at much smaller cost. We demonstrate the utility of these ideas by deriving elementary Monte Carlo algorithms for group membership and normal closure that have exponential reliability. These algorithms are appealing because they have worst case asymptotic running times that are better than the practical algorithms currently in use although not as good as algorithms that will be described later. Furthermore, these methods apply to the more general *black box groups* [4] and hence to matrix groups as well. The technique for incremental construction of subgroups is very effective when used in conjunction with a Monte Carlo algorithm for reducing the number of generators for a group (Theorem 2.9). An interesting application is a Monte Carlo algorithm for testing both solvability and nilpotency of a subgroup of $GL(n,q)$, the general linear group of dimension n over the Galois field $GF(q)$, which takes time polynomial in n and $\log q$. (All logarithms are taken to base 2.)

In Section 3, we describe a technique derived from random subproducts, referred to as *random prefixes*, which leads to a Monte Carlo normal closure algorithm for permutation groups of degree n that takes time $O(n^2 \log^4 n) = O^{\sim}(n^2)$. In general we use the notation $O^{\sim}(f(n))$ (read as "soft oh") to stand for $O(f(n) \log^c n)$ for some constant c.

In Section 4, we review recent work on membership algorithms for permutation groups. We first present two Monte Carlo algorithms which are based on rather different combinatorial ideas. The first is a general group membership algorithm with running time $O(n^3 \log^2 n)$, which relies heavily on the normal closure algorithm described in Section 3 and the primitivity structure of G [5]. The second algorithm is designed to be most effective on *small base* permutation groups. These are families of groups for which $\log |G| \leq \log^3 n$. This class of groups is particularly interesting since the permutation representations of non-alternating simple groups satisfyy this condition. For such groups, the algorithm



has running time $O^{\sim}(n)$, which is "nearly linear" in n. (A more precise time is given in Section 4.1 in terms of the base size.) The key ideas behind the small base algorithm are the "local-expansion" lemma of Babai [4] and a method for building "shallow" Schreier trees which is a constructive adaptation of a technique developed by Babai and Szemerédi [11] in a more theoretical context.

We complete our discussion of group membership by discussing a new unified approach which achieves superior asymptotic worst case running time in the general case. The simplest specialization to small base groups is not quite as good, although more other variations are currently under consideration. Nevertheless, this approach leads to a very simple algorithm which we believe has the potential for faster implementations.

Finally, we conclude in Section 5 with a discussion of base change algorithms for permutation groups. It is interesting to note that our original randomized base change algorithm [20] was developed for its own merit and only later proved crucial for the group membership algorithms of Sections 4.1 and 4.2. This experience was then reversed when combinatorial methods developed in the small base group membership algorithm were later found to be important for the cyclic base change algorithms [21].

2. Constructing Subgroups Through Random Elements

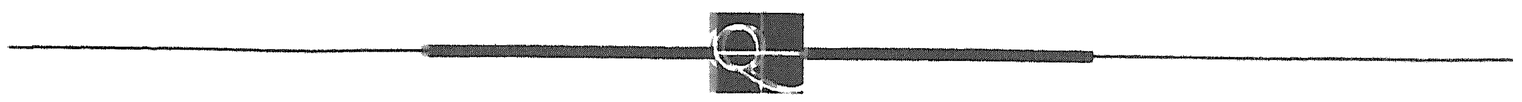
Many of our group computations involve the incremental construction of a subgroup H of G satisfying certain conditions. For example, H might be the subgroup stabilizing a point in a permutation representation of G, or the normal closure of a subgroup of G. Typically the process starts with an initial approximation H_0 to H, with H_0 the trivial group unless additional information is available. An increasing sequence of subgroups $\{H_i\}$ is then constructed with a generating set for H_i obtained through the addition of a single generator to a generating set for H_{i-1}. The expectation is that after a finite initial subsequence, all subgroups are equal to H. The most costly part of such computations is usually the selection of an element which, with positive probability exceeding some fixed $p > 0$, will strictly enlarge H_{i-1} to H_i whenever $H_{i-1} < H$. Suppose such a selection procedure is available and an upper bound L is known for the maximal length L of a subgroup chain of H. Then a standard application of Chernoff's bound [17] (see for example [26], Theorem 2.3, or [20], Theorem 3.5) yields the following inequality for constant c with $c > 1/p$.

Chernoff's bound: (specialized to subgroup chains)

$$\text{Prob}(H_{\lceil cL \rceil} = H) \geq 1 - \exp(-(1 - 1/pc)^2 pcL/2)$$

Random elements have this characteristic of enlarging a constructed subgroup with probability at least 1/2. However, truly random elements are usually achieved via construction of a strong generating set. *Random subproducts*, and variations thereof, share this characteristic of incrementally enlarging a subgroup with positive probability, while allowing one to avoid the costly intermediate step of constructing a strong generating set.

The fundamental theorem on random subproducts is presented, along with a



short proof, to demonstrate the simplicity of the underlying ideas.

DEFINITION. Given a generating set $S = \{g_1, \ldots, g_r\}$ for a group G, a random subproduct is an element w of the form

$$w = g_1^{e_1} g_2^{e_2} \cdots g_r^{e_r},$$

where $e_i = 0$ or 1 with probability 1/2.

PROPOSITION 2.1 (from [5]). *Let S generate a group G and let H be an arbitrary proper subgroup of G. Then a random subproduct w on S is not a member of H, with probability at least 1/2.*

PROOF. Let $S = \{g_1, \ldots, g_r\}$. There is a largest i for which $g_i \notin H$ (since $H \neq G$). So, w can be decomposed into the form $w = ug_i^{e_i}v$, with $g_i \notin H$ and $g_{i+1}, \ldots, g_r \in H$. Hence, $v \in H$.

We consider two cases. First, assume $u \in H$. With probability 1/2, $e_i = 1$, and so $w = ug_iv \notin H$. In the second case, assume $u \notin H$. With probability 1/2, $e_i = 0$, and so $w = uv \notin H$. This proves the proposition. □

This leads to a suite of algorithms for solving fundamental problems in computational group theory with dramatically reduced time and space complexity. These algorithms allow:

- constructions of a generating set of size $O(L)$ for a subgroup of H in G given a generating S for G and a transversal (defined in Section 2.1) for H in G using $O(L(|S| + [G : H]))$ multiplications [26].
- elementary normal closure of H in G where H and G have generating sets T and S respectively using $O(L(|S|+|T|))$ multiplications and inverses [26].
- reduction of a generating set S for G to size $O(L)$ using $O(|S| \log L)$ multiplications [5].
- normal closure using random prefixes where H and G are given by generating sets of size $O(L)$ using $O(L \log^4 L)$ multiplications [5].

The first two algorithms described above are based on simple extensions of random subproducts, *random Schreier subproducts* and *random normal subproducts*. In both cases, an element formed using these extensions has probability at least 1/4 of enlarging the subgroup being incrementally constructed. The random prefixes are a non-trivial variation of random subproducts, described in greater depth in the section on normal closure.

2.1. Random Schreier Subproducts: Simple Group Membership

The central problem for group membership is construction of a strong generating set. 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 subgroup* of G stabilizing α_i (notated G_{α_i}) is the subgroup $\{g \in G\colon\ {\alpha_i}^g = \alpha_i\}$, where ${\alpha_i}^g$ is the image of α_i in Ω under the permutation g. The *point stabilizer sequence* of G relative to α is the chain of subgroups

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

where $G^{(i)} = G_{\alpha_1,\ldots,\alpha_{i-1}}$, $1 \leq i \leq n$, is the subgroup of G consisting of all elements which fix each of the points α_j, $1 \leq j \leq i-1$. S is called a *strong generating set* for G relative to α if

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

Construction of a strong generating set allows one to efficiently construct a *(left) transversal* $T^{(i)}$ for a set of (left) cosets $G^{(i)}/G^{(i+1)}$ as i varies from 1 to n. For groups $H \subseteq G$, a left transversal of H in G is a set of left coset representatives of H in G. With a family of such transversals $\{T^{(i)}\}$ and with $g \in \mathrm{Sym}(\Omega)$, either one can express g as a factored word, $g_{n-1}g_{n-2}\cdots g_1$, for $g_i \in T^{(i)}$, or else conclude that $g \notin G$. This solves the group membership problem.

It is easy to show that given a group $G^{(i)}$ with generators $S^{(i)}$ and a transversal $T^{(i)}$ for $G^{(i)}/G^{(i+1)}$, the set of Schreier generators,

$$\{ug(\overline{ug})^{-1}\colon\ g \in S^{(i)}, u \in T^{(i)}, \overline{ug} \in T^{(i)}, \alpha_i^{ug} = \alpha_i^{\overline{ug}}\},$$

generates $G^{(i+1)}$.

While many previous algorithms used deterministic techniques to build a strong generating set, randomized techniques have recently been found to have faster complexities, and they also serve as an important technique in implementations. Let G be a finitely generated group with a generating set $S = \{g_1, g_2, \ldots, g_s\}$, let $H \subseteq G$ be a subgroup of finite index, and let $T = \{t_1, \ldots, t_n\}$ be a transversal of H in G. For $g \in G$, let $\overline{g}$ represent the unique element $t \in T$ such that $gt^{-1} \in H$. A *random Schreier subproduct* for H with respect to S and T is an instance of a product $(t_1g(\overline{t_1g})^{-1})^{e_1}(t_2g(\overline{t_2g})^{-1})^{e_2}\cdots(t_ng(\overline{t_ng})^{-1})^{e_n}$ where g is a random subproduct of S, and the e_i are independent random variables uniformly distributed over $\{0, 1\}$. The proof of the next result is elementary and is included for completeness.

PROPOSITION 2.2. *Let $G = \langle S \rangle$ for finite S, and let $H \subseteq G$ be a subgroup of finite index. Let T be a transversal for H in G. Then a random Schreier subproduct for H with respect to S and T can be computed at the cost of $O(|S| + |T|)$ group multiplications and inverses. Further,*

$$\mathrm{Prob}(h \notin K) \geq 1/4.$$

PROOF. Choose $\sigma \in T$ and i a maximal index such that $\sigma g_i(\overline{\sigma g_i})^{-1} \notin K$ for some $\sigma \in T$. (There exists such an i since $K \subseteq H$ is a proper subgroup, and the Schreier generators generate all of H.)

Given a random subproduct g on G, we show that with probability at least $1/2$ there exists a $\tau \in T$ such that $\tau g(\overline{\tau g})^{-1} \notin K$. Let u and v be prefixes and suffixes of g uniquely defined by $g = ug_i^{e_i}v$. The calculations below implicitly use the fact that $\overline{\overline{\tau}y} = \overline{\tau y}$ and hence $\overline{\tau}y(\overline{\overline{\tau}y})^{-1} = \overline{\tau}y(\overline{\tau y})^{-1}$. The argument divides into two cases. In the first case, $\tau u(\overline{\tau u})^{-1} \in K$ for all $\tau \in T$. So, with probability $1/2$, $e_i = 1$, $g = ug_iv$ (and so $u^{-1}g = g_iv$), and

$$\overline{\sigma u^{-1}}g(\overline{\sigma u^{-1}g})^{-1} = \underbrace{\overline{\sigma u^{-1}}u\sigma^{-1}}_{\in K}\ \underbrace{\sigma g_i(\overline{\sigma g_i})^{-1}}_{\notin K}\ \underbrace{\overline{\sigma g_i}v(\overline{\sigma g_i v})^{-1}}_{\in K} \notin K.$$

In the second case, there is a $\sigma' \in T$ such that $\sigma' u(\overline{\sigma' u})^{-1} \notin K$. So with probability $1/2$, $e_i = 0$, $g = uv$, and

$$\sigma' g(\overline{\sigma' g})^{-1} = \underbrace{\sigma' u(\overline{\sigma' u})^{-1}}_{\notin K}\ \underbrace{(\overline{\sigma' u})v(\overline{\sigma' uv})^{-1}}_{\in K} \notin K.$$

Let h be a random subproduct on $\{\tau g(\overline{\tau g})^{-1}\colon\ \tau \in T\}$. If $\tau g(\overline{\tau g})^{-1} \notin K$ for some $\tau \in T$, then $h \notin K$ with probability at least $1/2$. Since the exponents $\{e_i'\} \subseteq \{0,1\}$ for the random subproduct h are chosen independently of the $\{e_i\}$ for g, there is an overall probability of at least $1/4$ that $h \notin K$. □

We can apply Proposition 2.2 in conjunction with Chernoff's bound to obtain the following result.

THEOREM 2.3. *Let G be a group generated by a finite set S. Further, let L be an a priori upper bound on the length of subgroup chains in G, and let T be a transversal for $H \subseteq G$. Then one can construct a set of cL generators for H ($c > 4$) with probability at least $1 - \exp(-(1/4 - 1/c)^2 2cL)$ using at most $cL(|S| + 2|T|)$ group multiplications and inverses.*

For permutation groups, there is a bound $L < 3n/2$ on subgroup chains in S_n, determined by Cameron, Solomon and Turull [16] (and an earlier, coarser bound of $2n - 3$ by Babai [3]). Theorem 2.3 can be applied to obtain an elementary Monte Carlo group membership algorithm for a permutation group G which requires $O(n^4 + n^2|S|)$ time. We present a sketch of the argument. Initially, let $H = G^{(2)} = G_{\alpha_1}$ and let T be a right transversal for H in G. Applying Theorem 2.3, it follows that H can be generated by cn random Schreier subproducts with respect to S and T with probability at least $1 - \exp(-(1/4 - 1/c)^2 3cL)$. If we choose $c = 27$, then this probability will exceed $1 - \exp(-2n)$. Each random Schreier subproduct requires $O(n^2 + |S|)$ multiplications which dominates the running time. Iterating this step $n - 1$ times to obtain the successive point stabilizer subgroups, requires $O(n^2)$ multiplications per step. The probability of each step being correct simultaneously is easily seen to exceed $1 - \exp(-n)$. This leads to the following result.

COROLLARY 2.4. *Given a permutation group $G = \langle S \rangle$ of degree n, there is an elementary Monte Carlo algorithm for constructing a strong generating set for G which will require time $O(n^4 + n^2|S|)$ and has probability at least $1 - \exp(-n)$ of success.*

2.2. Random Normal Subproducts: Simple Normal Closure

Let $H = \langle U \rangle$ and $G = \langle S \rangle$ be finitely generated groups. Recall from the introduction that the normal closure of H under G is the smallest normal subgroup of $\langle H, G \rangle$ containing H. For H and G finite, one can easily develop an elementary construction in which a subgroup K is initially set to H, and conjugates k^g for $k \in K$ and $g \in G$ are successively added to K until closure is achieved. The k and g need only be drawn from generating sets for K and G, respectively.

A *random normal subproduct with respect to U and S* is a conjugate of the form $g^{-1}hg$ (notated h^g) where h is a random subproduct on U and g is a random subproduct on S. The fundamental result concerning random normal subproducts is given below. The proof is elementary.

PROPOSITION 2.5 (from [26]). *Let $H = \langle U \rangle$ and $G = \langle S \rangle$ for finite U and S. Suppose that $H^G \neq H$. Let h^g be a random normal subproduct with respect to U and S. Then*

$$\text{Prob}(h^g \in \langle H^G \rangle \setminus H) \geq 1/4.$$

This leads to a results which is analogous to Theorem 2.3.

THEOREM 2.6. *Let H and G be finite groups with $H = \langle U \rangle$ and $G = \langle S \rangle$. Let L be an a priori upper bound on the length of a chain of subgroups of $\langle H^G \rangle$. Then one can construct a set of cL generators ($c > 4$) for $\langle H^G \rangle$ with probability at least $1 - \exp(-(1/4 - 1/c)^2 2cL)$ using at most $cL(|S| + 2|T|)$ group multiplications and inverses.*

For permutation groups, one can choose $L = 3n/2$ and $c = 12$, yielding the following result.

COROLLARY 2.7. *Let H and G be permutation groups of degree n. Then a set of $12n$ random normal subproducts will generate $\langle H^G \rangle$ in time $O(n(|S| + |T|))$ with probability at least $1 - \exp(-n)$. In particular, if both S and T have size $O(n)$, then the running time is $O(n^3)$.*

2.3. Reduction of Generators

Reduction of generators is an important principle for the construction of subgroup chains, such as the point stabilizer sequence or the commutator series. Typically each subgroup in the chain is constructed using a generating set for its predecessor in the chain together with some defining condition. Without taking care to reduce the size of the generating set used for the construction, it is possible to have an exponential blowup in the number of generators as one precedes down the subgroup chain.

THEOREM 2.8 (from [26], Theorem 2.3). *Let $G = \langle S \rangle$ be a finite group. Let L be a known upper bound on the length of all subgroup chains in G. Then for any parameter p such that $0 < p < 1$, there is a constant c such that with probability p one can find a generating set S' with $|S'| = O(L)$, using $O(|S| c \log L)$ group operations.*

A further illustration of this principle is a Monte Carlo polynomial time test for solvability of a matrix group $G = \langle S \rangle$ with S a set of non-singular $n \times n$ matrices over $GF(q)$. The derived series of a group G is the chain of subgroups $G = G_0 \supseteq G_1 \supseteq G_2 \supseteq \cdots$ where $G_{i+1} = \langle [G_i, G_i] \rangle$ (the commutator subgroup). G is solvable if the series terminates in the trivial subgroup. If $H = \langle [G, G] \rangle$ and $G = \langle S \rangle$, then it is well-known that H can be constructed as the normal closure of $\langle \{[g_1, g_2]\colon\ g_1, g_2 \in S\} \rangle$ under G. This would suffice for an polynomial-time solvability test for matrix groups, except that the number of generators for each successive subgroup of the derived series could grow exponentially.

For a polynomial-time solvability test for unipotent matrices, the crude bound $L = n^2 \log q$ suffices. The algorithm alternates between reducing the size of the generating set (Theorem 2.8) and computing generators for the next group in the derived series via normal closure (Theorem 2.6). After L iterations, if each of the resulting generators is the identity, then the algorithm asserts that the group is solvable. Constants can be chosen so that the probability of error is $\Omega(1)$. Since each iteration takes time polynomial in n and $\log q$ and since there are at most L iterations with $L = n^2 \log q$, the overall test is polynomial in n and $\log q$. An analogous algorithm tests nilpotency using the descending central series.

PROPOSITION 2.9 (from [5]). *A subgroup of $GL(n, q)$ can be tested for solvability and nilpotency in the time for $O(L^3 \log L)$ multiplications and inverses, with $L = n^2 \log q$.*

The estimate of $L = n^2 \log q$ is asymptotically tight. To see this, consider the family of groups of unipotent matrices in $GL(n, q)$ for $q = p^r$ with p a fixed prime and q varying. Since it has order $q^{n(n-1)/2}$, it is a Sylow p-subgroup of order $p^{(\log_p q) n(n-1)/2}$. Hence for p fixed, $L = (\log_p q) n(n-1)/2 = \Theta(n^2 \log q)$ for that family.

3. NORMAL CLOSURE: RANDOM PREFIXES

An alternative normal closure algorithm running in time $O^{\sim}(n^2)$ for permutation groups of degree n is discussed next. The key idea is the use of random prefixes to produce new elements for the group under construction.

DEFINITION. A *random prefix* of a sequence of group elements $(g_1, g_2, \ldots, g_k)$ is an instance of a product of the first i elements, for i a random integer from 1 through k.

To see how random prefixes are used in the normal closure algorithm, assume

that the maximal length of a chain of subgroups of G is L and that G and H are each given by generating sets of size $O(L)$. Let S_G and S_H respectively of size $O(L)$. For a suitably large m ($m = \Omega(L \log^2 L)$), form lists S_G and T of length m initialized with the generating sets for G and H respectively and padded to the end with the identity element. Let $r = \Theta(\log(m))$ and let $\pi_1, \ldots, \pi_r$ be random permutations of $\{1, \ldots, m\}$ chosen at the beginning of the computation. Apply randomly chosen elements π_i and π_j to S_G and T respectively, let g be a random prefix of $S_G^{\pi_i}$ and h a random prefix of T^{π_j}. Replace one of the original identity elements of T by h^g. Then with fixed but arbitrarily high probability, after $O(mr^2)$ rounds, the list T will contain generators for $\langle H^G \rangle$. In Section 2.3, we showed how to reduce the generating set T to one of size $O(L)$.

The proof of correctness depends on being able to "spread out" the distance between two arbitrary elements in the list S_G or T using some permutation in S_m. A set of permutations having this property is called a set of ϵ-spreaders. The proof shows that the set of random permutations $\{\pi_1, \ldots, \pi_r\} \subseteq S_m$ is in fact a set of ϵ-spreaders. This relies on the construction of special permutations in S_m, called *ϵ-spreaders*. Given an arbitrary subset of generators, at least one of $\log L$ ϵ-spreaders is guaranteed to permute a list of m generators, so that the sum of distances of each element of the arbitrary subset to its nearest neighbor from the same subset is bounded below by ϵ. Efficient deterministic construction of ϵ-spreaders remains an important open question. An answer would allow a modified normal closure algorithm achieving both exponential reliability and a smaller time complexity.

THEOREM 3.1 (from [26], Theorem 3.5). *Let G and $H \subseteq G$ be finite groups and assume all subgroup chains in G have length at most L. Assume further that G and H are given by generating sets of size $O(L)$. Then one can construct $O(L)$ generators for $\langle H^G \rangle$ with fixed but arbitrarily high probability using $O(L \log^4 L)$ group operations.*

In the original Theorem 3.5 in [5], the time was reported as $O(L \log^3 L)$, whereas a more careful counting seems to show it as $O(L \log^4 L)$. In the case of permutation groups, $L = O(n)$ and we have the following corollary.

COROLLARY 3.2 (from [26], Theorem 3.5). *Let H and G be permutation groups of degree n with $H \subseteq G$. Assume that both H and G have generating sets of size at most $O(n)$. Then a generating set of size $O(n)$ for $\langle H^G \rangle$ can be obtained with fixed but arbitrarily high probability using $O(n \log^4 n)$ group multiplications and inverses. If G is drawn from a family of small base groups (for which $\log |G| = O^\sim(1)$), then $O^\sim(1)$ such group operations are required.*

4. GROUP MEMBERSHIP

From a complexity viewpoint, group membership is probably the problem most studied in computational group theory. Originally, implementations of the formal algorithms were not competitive with a straightforward heuristic algorithm such as randomized Schreier-Sims. We now have deterministic [22] and

Monte Carlo implementations [6] using new ideas with provable complexities that are competitive in implementations (see also [32]). Nevertheless, there is considerable hope that further study of the new "building blocks" for group membership will simultaneously satisfy the goal of having fast implementations which have provable reliability and lower complexity.

The notion of a base, along with the first polynomial-time group membership algorithm, was introduced by Sims [33]. A base for a permutation group G acting on Ω is a subset $B \subseteq \Omega$ such that $g \in G$ is the identity if and only if $\beta^g = \beta$ for all $\beta \in B$. This implies that $\log |G| \leq |B| \log n$. The base B is non-redundant, if any proper subset of B is not a base. We will say that $B = \{\beta_1, \ldots, \beta_k\}$, considered as a sequence, is non-redundant with respect to the given ordering if $G_{\beta_1,\ldots,\beta_{i-1}} \neq G_{\beta_2,\ldots,\beta_i}$ for all $1 \leq i \leq k$. If B is non-redundant (with respect to any ordering), then

$$|B| \leq \log |G| \leq |B| \log n.$$

We tend to view the group membership problem as one that requires several algorithmic techniques depending on the size of the input group. A family of permutation groups $\mathcal{G}$ is a *small base family* if there is a constant c such that for any group $G \in \mathcal{G}$ with degree n, there is a base for G of size $M \leq \log^c n$. Alternatively, a small base family of groups can be characterized as satisfying a bound $|G| \leq O(\log^{c+1} n)$ for some constant c and for all $G \in \mathcal{G}$. In practice, we may choose $c = 2$. In this case, the family of small base permutation groups includes all permutation representations for the non-alternating simple groups [15] (often base size less than 8 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. This can be seen from the fact that a strong generating set for G relative to a base B, will always require $\Omega(Mn)$ storage, assuming that B is non-redundant. Although we casually refer to the remaining groups as *large base* permutation groups, further refinements are possible but will not be discussed here.

4.1. Small Base Group Membership

An algorithm was presented for computing a strong generating set [6], which proves the following result.

THEOREM 4.1. *Let $G = \langle S \rangle \leq \mathrm{Sym}(\Omega)$, $|\Omega| = n$, $|S| = O(\log |G|)$, and suppose that the maximal size of a non-redundant base with respect to any ordering is $b = O(\sqrt[3]{n/\log n})$. Then, with fixed but arbitrarily high probability, a strong generating set for G can be constructed in $O(nb \log^3 |G| \log b) = O(n^{7/3} \log^{8/3} n)$ time. The constructed strong generating set supports membership testing in $O(n \log |G|)$ time and the memory requirement is $O(n \log |G|)$.*

This algorithm is referred to as the *small base group membership algorithm*, because it is designed to be most effective on groups with a small base. For such groups, $\log |G| \leq \log^3 n$ and so the algorithm runs in nearly linear time. Of course, this information is not known a priori and the algorithm makes no assumptions about the nature of G.

We now briefly describe the key ideas which lead to the removal of the quadratic bottlenecks ($O(n^2)$ terms in the timing analysis) for the small base group membership algorithm.

4.1.1. Local Expansion in Combination with Random Subproducts

The following situation occurs in the usual "bottom-up" construction of a strong generating set for G. Given a point $x \in \{1, 2, \ldots, n\}$ and a known strong generating set for a subgroup H of G_x, test if $H = G_x$. This usually requires sifting $O(n|S|)$ Schreier generators where S is a generating set for G. Each sift takes time $O(n)$ and so the test requires $O(n^2|S|)$ time. In effect, one is testing for strong generation the set of generators consisting of the union of S with generators for H.

The "local expansion lemma" for groups [4] is the basis for an efficient strong generating test for permutation groups [6] which requires only $O^{\sim}(|S|n)$ time for small base groups with generators S. It states that any subset D of a group G which consists of words of length $\leq t$ in the generators S and satisfying $|D| \leq |G|/2$, also satisfies

$$|Dg - D|/|D| \geq 1/(4t)$$

for at least one generator $g \in G$. The constant 4 was recently improved to 1 in [10], under certain technical assumptions.

The principle of the strong generating test is to identify D with all factored words of G in S. If a data structure is available (such as short or cube Schreier trees defined in the next section) for extracting the "known" coset representatives of $G^{(i)}/G^{(i+1)}$ as a word of depth $O^{\sim}(1)$, and if the currently "known" base is of size $O^{\sim}(1)$, then t can be bounded by $O^{\sim}(1)$. So if $|D| \leq |G|/2$, there is a $g \in S$ such that for random $d \in D$, $dg \notin D$ with probability at least $1/4t$. This test is repeated for each point stabilizer subgroup $G^{(i)}$ and strong generating set $G^{(i)} \cap S$.

Random subproducts further improve the strong generating test [6], Lemma 3.4. Instead of testing if $dg \in D$ for random $d \in D$, and for each $g \in G$, it suffices to test dw for w a random subproduct of the generating set S.

4.1.2. Schreier Trees

A critical choice in algorithm design is the choice of a data structure for representing coset representatives of G_x in G. *Schreier trees* are often used for their space-efficiency. A *Schreier tree* for G/G_x is a directed labeled tree (T, S) where T is a tree with root x, nodes x^G and edge labels chosen from S. Coset representatives are obtained by taking the product of edge labels along a path from the root to the specified node. Schreier trees require only $O(|S|n)$ space, but the cost of computing a single coset representatives can take time $O(n^2)$ in the case of trees of depth n. This condition can hold even for small base groups (e.g. consider the Schreier tree constructed from a cyclic group of order n).

Ideally, one would like to build *short* Schreier trees, i.e., ones with depth $O(\log n)$ using $O(\log n)$ edge labels. As part of our work on a fast random base

changed [24], we showed how to construct short Schreier trees when random elements are available. This is always the case, in the context where one is performing a base change. The following observation is the key to short Schreier trees. Although we discovered it independently, other authors have used variations of it from as early as 1965 for a variety of results ([1], [7], [27]). Most recently, it was used as part of Babai's result on local expansion [4].

Let G act transitively on Ω with $|\Omega| = n$. For fixed $P \subseteq \Omega$ and random $g \in G$,

$$E(|P^g - P|) = |P|(n - |P|)/n,$$

where E is the expectation over $g \in G$. We identify P with the current nodes of a partially built Schreier tree. As long as fewer than half of the elements of Ω are in the tree, each random element has a fixed positive probability of expanding the tree by a constant factor. Thereafter, with fixed positive probability, each random element will decrease by a constant factor the number of nodes of Ω not yet in the tree. It is still an open problem to deterministically build short Schreier trees (depth $O(\log n)$) efficiently. This would eliminate the need for a source of random elements, which generally are not present in the course of constructing a strong generating set.

The key step for the small base group membership algorithm was a constructive adaptation of a result originally developed [11] in a purely theoretical context, to deterministically build *cube Schreier* trees of depth $O(\log |G|)$.

Given a sequence $R = (g_1, \ldots, g_r)$, define the cube over R as

$$C(R) = \{g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_r^{\epsilon_r} \colon \epsilon_i \in \{0, 1\}\}.$$

It is easy to see [11] that for $g \in G$

$$g \notin C(R)^{-1}C(R) \iff |C(R \cup \{g\})| = 2|C(R)|.$$

Choosing such a g is made efficient by noting that

$$1^g \notin 1^{C(R)^{-1}C(R)} \Rightarrow g \notin C(R)^{-1}C(R).$$

Thus, while $1^{C(R)^{-1}C(R)} \neq \Omega$, we can double the size of the cube $|C(R)|$ by appending a g such that $1^g \notin 1^{C(R)^{-1}C(R)}$. After at most $\log_2 |G|$ steps, we have $1^{C(R)^{-1}C(R)} = \Omega$. A Schreier tree with depth at most the twice the length of R (at most $2 \log_2 |G|$) can then be built in the obvious way.

A slightly more subtle way of building Schreier trees which preserves the theoretical guarantee that the depth will never exceed $2 \log_2 |G|$, while working fast in practice, is given by the following code.

Procedure *Build-Cube-Schreier-Tree*(S, x)
Input: A generating set S for G and point $x \in \Omega$.
Output: A cube Schreier tree $\mathcal{T}$ for x^G with the property that $\texttt{Labels}(\mathcal{T}) = R \cup R^{-1}$, where $C(R)$ is a non-degenerate cube and $\texttt{depth}(\mathcal{T}) \leq 2|R|$.

Initialize $R \leftarrow ()$
Set $\texttt{root}(\mathcal{T}) \leftarrow \{x\}$, $\texttt{Labels}(\mathcal{T}) \leftarrow ()$
  While there exists $g \in S$ such that
    $\texttt{Nodes}(\mathcal{T})^g \neq \texttt{Nodes}(\mathcal{T})$ do

Let $y \in \texttt{Nodes}(\mathcal{T})$ such that $y^g \notin \texttt{Nodes}(\mathcal{T})$
Let $h =$`Coset-rep`$(\mathcal{T}, y)$
Append hg to R [Note $x^{hg} \notin \texttt{Nodes}(\mathcal{T})$]
Build a new $\mathcal{T}$ using breadth first search with $R \cup R^{-1}$ to level $2|R|$
Return$\{\mathcal{T}\}$

Using this procedure, it is possible to prove the following result.

PROPOSITION 4.2 (from [21]). *Given a generating set S for G (not necessarily strong) and an ordering $\alpha = (\alpha_1, \ldots, \alpha_n)$ of $\{1, 2, \ldots, n\}$, it is possible to build Schreier trees for $\langle S \cap G_{\alpha_1,\ldots,\alpha_{i-1}} \rangle$, with depth at most $2 \log |G^{(i)}|$, $1 \leq i \leq n$, in overall time $O(n \log^2 |G|)$.*

4.2. Large Base Group Membership Algorithm

A group membership algorithm was described in [5] which leads to the following result.

THEOREM 4.3 (from [5]). *Given $O(n^2 \log n)$ generators of a finite permutation group G acting on n points, one can construct a strong generating set for G in $O(n^3 \log^2 n)$ Monte Carlo time.*

The algorithm used to prove Theorem 4.3 will be referred to as the *large base* group membership algorithm. In contrast to the previously fastest algorithm [8], this new algorithm does not require results from the classification of finite simple groups. The algorithm does run faster if appeal is made to the result from the classification that only S_n and A_n are 6-transitive, although this can be eliminated without affecting the asymptotic running time by appealing to the "Fast-Giant" routine described in [8]. The other key results which play a crucial role in obtaining this result are the reduction of generators result described in Section 2.3 and the $O^{\sim}(n^2)$ normal closure result described in Section 3. The algorithm outputs a data structure which supports membership testing and the construction of random elements but not with respect to the traditional point stabilizer sequence. In order to convert to the traditional strong generating set, one may employ the random base change algorithm described in [20] and referred to in Section 5.

4.3. Unified Group Membership Algorithm

The authors have attempted to derive a unified approach to both small and large base group membership algorithms [26]. The results are partially successful in that essentially the same approach leads to a $O^{\sim}(n^3)$ group membership algorithm for general groups and a $O^{\sim}(n)$ group membership algorithm for small base groups. However, as of this writing, the specialization to large base groups leads to an improvement of Theorem 4.3, while the specialization to small base groups leads to a timing analysis that is not yet as good as the one given in Theorem 4.1. Nevertheless, the basic approach is extremely simple and has the potential for leading to superior implementations.

The key idea is a variation on the traditional sifting routine, which uses cube

Schreier trees. The global data structure $S^{(i)}$ represents a set of group elements of $G^{(i)}$ to be constructed, such that $\cup_{j=i}^{n} S^{(j)}$ generates $G^{(i)}$. $C^{(i)}$ is the cube over $S^{(i)}$. The cube $D^{(i)} = C^{(n-1)}C^{(n-2)}\cdots C^{(i)}$ is derived from $C^{(i)}$, and is used purely as a notational convenience to describe the algorithm. One need not store any data structure corresponding to $D^{(i)}$. The points $\{\alpha_i\}$ are the permutation domain. Sets such as $\alpha_i^{C^{(i)}}$ and $\alpha_i^{D^{(i)-1}D^{(i)}}$ are re-computed whenever needed (although caching schemes are possible).

Procedure *Deep-Sift*(i, g)
Input: $1 \leq i \leq n-1$; $g \in G^{(i)}$

Output: Either a level $i' \geq i$ and $g' \in G^{(i')}$ such that $|\alpha_{i'}^{C^{(i')}\{g',1\}}| \geq 2|\alpha_{i'}^{C^{(i')}}|$ or else "fail";
"fail" occurs if and only if $g \in D^{(i)-1}D^{(i)}$
Time: $O(n \log |G|)$
If $|\alpha_i^{C^{(i)}\{g,1\}}| \geq 2|\alpha_i^{C^{(i)}}|$ then [add g to strong generating set]
Set $S^{(i)} \leftarrow S^{(i)} \cup \{g\}$
Set $C^{(i)} \leftarrow C^{(i)}\{g, 1\}$ Return(i, g)
Else if $i = n-1$ then
Return("fail")
Else [In this case, $|\alpha_i^{C^{(i)}g} \cap \alpha_i^{C^{(i)}}| \neq \emptyset$]
Set $u \leftarrow$ arbitrary element of $C^{(i)}$ such that ${\alpha_i}^u \in {\alpha_i}^{C^{(i)}g^{-1}}$
Set $u' \leftarrow$ arbitrary element of $C^{(i)}$ such that ${\alpha_i}^{ug} = {\alpha_i}^{u'}$
Return(*Deep-Sift*$(i+1,\ ugu'^{-1})$)

Verifying the output property of *Deep-Sift* reduces to demonstrating that the property $g \notin D^{(i)-1}D^{(i)}$ is preserved in recursive calls. To see this, note that

$$\begin{aligned} g \notin D^{(i)-1}D^{(i)} &\Rightarrow D^{(i)}g \cap D^{(i)} = \emptyset &&\Rightarrow D^{(i+1)}ug \cap D^{(i+1)}u' = \emptyset \\ &\Rightarrow D^{(i+1)}ugu'^{-1} \cap D^{(i+1)} = \emptyset &&\Rightarrow g \notin D^{(i+1)-1}D^{(i+1)}. \end{aligned}$$

So $g \in D^{(i)-1}D^{(i)}$ or else $|\alpha_{i'}^{C^{(i')}\{g',1\}}| = 2|\alpha_{i'}^{C^{(i')}}|$ for some arguments to a recursive call, i' and g'. The main point to observe is that in the course of attempting to build a strong generating set, there can be at most $\log |G|$ successful calls to *Deep-Sift*.

The main algorithm for constructing a strong generating set is fashioned from *Deep-Sift* using a top-down approach. In the case of a general group membership algorithm random Schreier subproducts are used. For the special case of small base groups, the principle of local expansion is applied. The results for both cases are summarized in the next theorem.

THEOREM 4.3(from [26]). *Given a permutation group specified by a generating set S, a strong generating set relative to any ordering can be computed in Monte*

Carlo time $O(n^2 \log |G| \log n + n|S| \log |G|)$ *time with reliability at least* $1 - 1/n$ *and with space requirements* $O(n \log |G|)$. *With suitable modifications for the special case of a small base, a strong generating set can be computed in Monte Carlo time* $O(nb^2 \log^2 |G| \log n + n|S| \log |G|)$ *with the same reliability and space requirements, where* b *is the size of a maximal non-redundant base with respect to any ordering.*

4.4. Implementation Notes

A completely faithful implementation of the algorithms described in Sections 4.1 and 4.2 would be both slower than optimal in practice and complex. The complication can be estimated by the existing implementations. We have written straightforward 1,500 and 1,000 line implementations respectively in our LISP system. Seress and Weisz have separately written a 2,000 line standalone C version (plus 2,000 lines for user-friendly I/O) for the (small base) algorithm in which they "fine-tuned" the numerous parameters for optimal performance.

Thus, simplifying the algorithms is an important research goal. We believe that such simplifications are also likely to yield faster implementations. We believe the approach described in Section 4.3 will ultimately lead to algorithms which are simpler to program and have superior performance. However, this has not yet been demonstrated and is still under active investigation.

5. Base Change

Let G be a subgroup of S_n and let S be a strong generating set for the point stabilizer sequence of G relative to an ordering $\alpha = \alpha_1, \ldots, \alpha_n$ of $\{1, 2, \ldots, n\}$. A *change of base* is the construction of a strong generating set S' for G relative to a new ordering α'. Base change is a crucial algorithm in many important group computations and plays an especially important role in many of the backtracking algorithms currently used to solve problems for which provably efficient algorithms are not currently known.

In [24], we presented a random base change algorithm which uses $O(n \log^2 |G|)$ Las Vegas time and $O(n \log |G|)$ space. (A Las Vegas algorithm provides a deterministic guarantee of correctness, but not of running time.) For large base groups, the complexity of this algorithm is inferior to the deterministic $O(n^3)$ base change algorithm [12]. However, it has almost linear complexity for small base groups. The key to deriving this result is the construction of short Schreier trees (described in Section 4.1.2). Interestingly enough, both the large base and small base group algorithms rely heavily on the random base change algorithm, but for different reasons. The small base group algorithm requires the existence of short Schreier trees and the large base group algorithm requires a base change after completion in order to realign the original point set.

A *cyclic base change* occurs when α is obtained from α' through a *right cyclic shift.* In this case, α and α' satisfy the relationship

$$\begin{aligned} \alpha &= \alpha_1, \ldots, \alpha_r \ldots, \alpha_{s-1}, \alpha_s, \ldots, \alpha_n \\ \alpha' &= \alpha_1, \ldots, \alpha_s, \alpha_r, \ldots, \alpha_{s-1}, \ldots, \alpha_n \end{aligned}$$

It is this special case of base change that is required in many backtracking algorithms ([13], [14], [30]). It was shown in [12] that a deterministic cyclic base change could be performed in time $O(n^2)$ using $O(n^2)$ space.

The next series of results concerns the important case of a cyclic base change.

THEOREM 5.1. *Given a strong generating set S for G, a deterministic cyclic base change algorithm can be described which requires $O(n \log^2 |G| + n|S| + n \log n)$ time and $O(n \log |G|)$ space.*

The key to the proof of Theorem 5.1 is the technique described in [12] in conjunction with Proposition 4.2. Randomized methods are the key to then next theorem, which substantially improves Theorem 5.1 under a hypothesis of availability of fast generation of random group elements. This is satisified if a *short Schreier vector data structure* is available.

THEOREM 5.2 (from [21], Theorem B). *Assume that random elements can be computed in time $O(n \log |G|)$. Let b be the size of a base relative to some ordering α and let α' be an ordering obtained from α by a right cyclic shift. Then 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 proof of Theorem 5.2 depends on the following observation: if H is a subgroup of a finite group G, U a complete set of right coset representatives for H in G and $S \subseteq G$ 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 (for $g \in G$, $\bar{g}$ is the unique element of U so that $H\bar{g} = Hg$). This observation is applied in the context of sifting down a subgroup chain of a permutation group G. The required subgroup chain is defined by the problem of performing a right cyclic shift. Initially, we have a set S of $c \log n$ mutually independent elements of G for some constant c. As we move down the subgroup chain, each element is multiplied by a suitable coset representative to produce a set of $c \log n$ mutually independent random elements for the next subgroup in the chain. These $c \log n$ random elements then suffice to build a short Schreier tree for a suitable orbit of the subgroup. The proof of the last statement is in Theorem 3.5 of [24].

REFERENCES

1. D. ALDOUS (1987). On the Markov Chain Simulation Method for Uniform Combinatorial Distributions and Simulated Annealing. *Probability in Engineering and Informational Sciences* 1, pp. 33–46.
2. L. BABAI (1979). Monte-Carlo Algorithms in Graph Isomorphism Testing. *Université de Montréal Dep. Math. et Stat., Tech. Report D.M.S.* 79–10.
3. L. BABAI (1986). On the Length of Chains of Subgroups in the Symmetric Group. *Comm. in Algebra* 14, pp. 1729–1736.

4. L. Babai. Bounded Round Interactive Proofs in Finite Groups. *SIAM J. Discr. Math.*, to appear.
5. L. Babai, G. Cooperman, L. Finkelstein, E.M. Luks, and Á. Seress (1991). Fast Monte Carlo Algorithms for Permutation Groups. *Proc. 23rd ACM STOC*, pp. 90–100.
6. L. Babai, G. Cooperman, L. Finkelstein, and Á. Seress (1991). 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.
7. L. Babai and P. Erdős (1982). Representation of Group Elements as Short Products. *Annals of Discrete Mathematics* 12, pp. 27–30.
8. L. Babai, E. Luks and Á. Seress (1988). Fast Management of Permutation Groups. *Proc. 29th IEEE FOCS*, pp. 272–282.
9. L. Babai and Á. Seress (1987). On the Degree of Transitivity of Permutation Groups: A Short Proof. *J. Combinatorial Theory: Series A* 45(2), pp. 310–315.
10. L. Babai and M. Szegedy (1992). Local Expansion in Symmetrical Graphs. *Combinatorics, Probability and Computing* 1, to appear
11. L. Babai and E. Szemerédi (1984). On the Complexity of Matrix Group Problems I. *Proc. 25th IEEE FOCS*, Palm Beach, FL, pp. 229–240.
12. C.A. Brown, L. Finkelstein and P.W. Purdom (1989). A New Base Change Algorithm for Permutation Groups. *SIAM J. Computing* 18, pp. 1037–1047.
13. G. Butler (1982). Computing in Permutation and Matrix Groups II: Backtrack Algorithm. *Math. Comp.* 39, pp. 671–680.
14. G. Butler and C. Lam (1985). General Backtrack Algorithm for the Isomorphism Problem of Combinatorial Objects. *J. of Symbolic Computation* 1, pp. 363-381.
15. P.J. Cameron (1981). Finite Permutation Groups and Finite Simple Groups. *Bull. London Math. Soc.* 13, pp. 1–22.
16. P.J. Cameron, R. Solomon and A. Turull (1989). Chains of subgroups in symmetric groups. *J. of Algebra* 127, pp. 340–352.
17. H. Chernoff (1952). A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Math. Statistics* 23, pp. 493–507.
18. G. Cooperman and L. Finkelstein. New Methods for Using Cayley Graphs in Interconnection Networks. *Discrete Applied Mathematics*, Special Issue on Interconnection Networks, in press.
19. G. Cooperman and L. Finkelstein (1991). A Strong Generating Test and Short Presentations for Permutation Groups. *J. Symbolic Computation* 12, pp. 475–497.
20. G. Cooperman and L. Finkelstein. A Random Base Change Algorithm for Permutation Groups. *J. Symbolic Computation*, to appear.
21. G. Cooperman and L. Finkelstein. Fast Cyclic Base Change for Permu-

tation Groups. *Proceedings of the International Symposium on Symbolic and Algebraic Computation* (ISSAC 92), to appear.

22. G. COOPERMAN, L. FINKELSTEIN and P.W. PURDOM (1989). Fast Group Membership using a Strong Generating Test for Permutation Groups. *Computers and Mathematics*, (E. KALTOFEN and S.M. WATT, eds.) Springer-Verlag, New York, pp. 27–36.
23. G. COOPERMAN, L. FINKELSTEIN and E. LUKS (1989). Reduction of Group Constructions to Point Stabilizers. *Proceedings of the International Symposium on Symbolic and Algebraic Computation* (ISSAC 89), ACM Press, pp. 351–356.
24. G. COOPERMAN, L. FINKELSTEIN and N. SARAWAGI (1990). 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, pp. 161–168.
25. G. COOPERMAN, L. FINKELSTEIN and B. YORK (1990). A Parallel Implementation of Group Membership and the Method of Random Subproducts. Northeastern University Technical Report NU-CCS-90-17.
26. G. COOPERMAN, L. FINKELSTEIN (1992). Combinatorial Tools for Computational Group Theory, submitted to the Proceedings of the Dimacs Workshop on Groups and Computation.
27. P. ERDŐS and A. RÉNYI (1965). Probabilistic methods in group theory. *J. d'Analyse Math.* 14, pp. 127–138.
28. W.M. KANTOR (1992). Notes on Polynomial-Time Group Theory. This issue of CWI Quarterly.
29. J. LEON (1980). On an Algorithm for Finding a Base and Strong Generating Set for a Group Given by a Set of Generating Permutations. *Math. Comp.* 35, pp. 941–974.
30. J. LEON (1984). Computing Automorphism Groups of Combinatorial Objects. *Computational Group Theory*, edited by M.D. ATKINSON, Academic Press, pp. 321–337.
31. P.M. NEUMANN and C. PRAEGER (1991). A Recognition Algorithm for Special Linear Groups. *Mathematics Research Report* 008-91, Australian National University.
32. Á. SERESS and I. WEISZ (1992). PERM: A Program Computing Strong Generating Sets. Submitted to the Proceedings of the Dimacs Workshop on Groups and Computation.
33. C.C. SIMS (1971). Computation with Permutation Groups. *Proc. Second Symposium on Symbolic and Algebraic Manipulation*, edited by S.R. PETRICK, ACM Press, pp. 23–28.
34. H. WIELANDT (1964). *Finite Permutation Groups*. Academic Press, New York.