Fast Management of Permutation Groups

László Babai
Eugene M. Luks
Ákos Seress

CIS-TR-88-13
August 3, 1988

Department of Computer and Information Science
University of Oregon

To appear in *Proc. 29th Ann. FOCS Symp.* (1988)

Fast Management of Permutation Groups

László Babai *
Eötvös University, Budapest,
and University of Chicago

Eugene M. Luks †
University of Oregon

Ákos Seress ‡
Mathematical Institute of the
Hungarian Academy of Sciences
and Ohio State University

Abstract

We present new algorithms for computation in permutation groups. These enable an order-of-magnitude improvement in the worst-case analysis of the basic permutation-group problems, including membership-testing and computing the order of the group. For deeper questions about the group, including finding composition factors, we realize an improvement of up to four orders of magnitude. These and other essential investigations are all accomplished in $O(n^4 \log^c n)$ time.

The new machinery is distinguished by its recognition and use of the intrinsic structure of the group at hand. Notably, this comes into play precisely when the routine divide-and-conquer of the set (i.e., orbits, imprimitivity blocks) bottoms out (at primitive groups). The structure of large primitive groups allows an augmentation of the underlying set that readmits naive decomposition of the problem. As a result, the new base cases are restricted to "small" groups or full alternating groups. Another advance is our handling of the latter where, in addition to speeding up known methods by an order of magnitude, we eliminate yet an additional n^2 in a superfast Las Vegas algorithm for this frequent subcase.

Our timing analyses depend, in several ways and probably unavoidably, on consequences of the Classification of Finite Simple Groups.

1. Introduction

Since the size of a permutation group G on n elements can be exponential in n, it is customary, and usually necessary, to specify G by a small set S of generating permutations. The issue then arises as to whether the succinctness of such representation admits computationally efficient solutions to basic queries about G. The very fundamental question of deciding "Is $\sigma \in G$?" was not observed to be in P until [FHL]. Indeed, even NP may not seem obvious, since the minimal word in S that represents σ could have exponential length. Additionally dissuasive was the fact that Kozen [Ko] had shown membership in transformation *semigroups* is complete for PSPACE.

As it turned out, an efficient solution to permutation-group membership was already implemented in the computational-group-theory library in elegant routines devised by Sims [Si70]. In effect, Furst, Hopcroft, and Luks [FHL] demonstrated that a variant of Sims' algorithm runs in polynomial-time. The basic tool is a *strong generating set* (SGS). An SGS is comprised of coset representatives for each group modulo its successor in the subgroup tower

$$G = G_0 \geq G_1 \geq \cdots \geq G_{n-1} = 1$$

where G_i denotes the subgroup of G that fixes the first i points in a specified ordering of the set. (This is not precisely Sims' usage, see section 3.4). With respect to an SGS, representations of group elements are guaranteed to be short and, in fact, easily computable by *sifting* through successive coset tables. Thus, given an SGS, membership-testing can be accomplished in $O(n^2)$ time per test. The main result in [FHL] is an $O(n^6)$-time construction of an SGS.

Subsequently, Knuth [Kn] described a faster SGS construction, which, employing an elementary number-theoretic estimate on the length of subgroup chains in permutation groups, he analyzed to run in time $O(n^5 \log \log n)$. A new estimate by Babai [Ba83] (cf. [Ba86a]) of the length of such chains established that Knuth's algorithm actually runs in time $O(n^5)$. In the meanwhile, another variant, devised by Jerrum [Je82], was shown to run in $O(n^5)$. Jerrum also introduced a trick for reducing the space requirement from $O(n^3)$ to $O(n^2)$. (Our faster procedures can also be implemented within this space bound). Since Knuth's 1981 note and Jerrum's 1982 paper, the $O(n^5)$ bound has remained intact and has achieved some notoriety. It seems the limit for the established approach.

*Research partially supported by NSF Grant CCR-8710078 and Hungarian Natl. Found. for Scient. Res. Grant 1812.

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

‡Research partially supported by Hungarian Natl. Found. for Scient. Res. Grant 1812.

The methods in this paper introduce a number of entirely new principles.

One key departure from tradition is our construction and use of another sort of subgroup tower, one which is not easily observable solely in terms of the action on the set. Its very specification subsumes knowledge of the group structure. Nevertheless, for efficient computation (and ease in implementation), we are able to describe an elementary augmentation of the set, so that the tower can be reinterpreted as a sequence of point-stabilizers. The "new" points are nodes in a *structure graph* ($\mathcal{SG}$) This is a derivative of the familiar *structure forests* of [BL], [Re], [GHLSW] which display the orbit/imprimitivity-block breakdown of the underlying set (but, mere use of those forerunners does not lead to any speed-up). In particular, the $\mathcal{SG}$ draws out essential, but previously obscure, normal subgroups; these now emerge within the new point-stablizer chain as subgroups that fix the points through $\mathcal{SG}$ "levels".

The computation of the necessary point-stabilizing subgroups involves a second critical departure from conventional methods. The generation of each successive group via elements "sifted" from its predecessor, as in the previous SGS methods (see [FHL]), would still require $O(n^4)$ sifts each costing $O(n^2)$. Instead, we break the tower into stages, delimited by the aforementioned normal subgroups. We speed up the completion of the groups in a single stage via our introduction of a process that we call *normal sifting*; in this, a relatively small number of elements need to be sifted through an SGS for G/N to produce *normal generators* of N (i.e., generators of a subgroup whose normal closure is N). This is, more precisely, a *class* of procedures, for the choice of sifts can be linked to the nature of the group G/N. Basically, we are at liberty to sift the relations in any concise group presentation. (To exploit this, we reveal a fast construction of short presentations).

A third, quite radical, divergence from earlier methods occurs in our handling of full symmetric or alternating groups. Note, as we indicate in section 6.1, we need a constructive approach to membership (as opposed to "yes/no"). Even for these groups, the problem previously required $O(n^5)$ time. A technique for reducing that bound is inspired by methods of [BS].

We should also emphasize a fourth ingredient that is introduced in the new methods. Earlier procedures for the basic problems rely on naive group theory. For their development and understanding it suffices to cite Lagrange's Theorem (that the order of a subgroup divides the order of the group). On the other hand, to justify our new approach, we appeal to consequences of the classification of finite simple groups ($CFSG$). (However, no specific knowledge of this work is required of the reader; the required consequences have elementary formulations). Note that, for the basic problem of membership-testing, $CFSG$ is essential only to justify the timing, not the correctness; in particular, overstepping would provide a constructive counterexample to a consequence of $CFSG$.

This study serves as a companion to [BLS] in which a similar foundation is established for parallel (NC) management of permutation groups. In fact, for concise presentation herein, we sometimes direct the reader to [BLS] for details of an analogue. We warn, however, that the earlier paper should not be viewed as a source of efficient sequential algorithms. To be sure, a knowledgeable sequential implementation of the relevant portions of [BLS] results only in an $O(n^6)$ procedure for membership-testing, not even matching the previous record. Rather, the contribution of [BLS], in conjugation with the present paper, is a framework for dealing with permutation groups, one that has admitted many new techniques, both sequential and parallel.

We enumerate the new results in section 2. Section 3 gives definitions, some basic structures, and a brief exposition of Knuth's organization of Sims' algorithm. The major steps in the main algorithm are indicated in section 4. A divide-and-conquer procedure in section 5 obviates consideration of any "large" permutation groups except the alternating group. The key ingredient for treating alternating groups is given in section 6. Section 7 describes normal sifting, which is employed in passing down the levels of the $\mathcal{SG}$. The treatment of a given level is indicated in section 8, which separates the "small" group case from the alternating group case. A hint at our improvement of the composition factors algorithm is in section 9. Finally, in section 10, we review the obstructions to achieving an additional order of magnitude improvement; there are essentially three, two if one seeks only Las Vegas solutions, and only one in a Monte-Carlo procedure.

2. Main results

We survey the major new results. Throughout, n denotes the size of the permutation domain.

An SGS can be viewed as a data structure that supports membership testing in $O(n^2)$ time per test. We insist on the same for the generalized SGS that we construct in this paper. Letting s denote the size of a given list S of generators, the Knuth and Jerrum procedures for producing such a data structure run in time $O(n^5 + sn^2)$. Our central result is

Theorem 2.1. *Given a permutation group G by a list S of generators, $|S| = s$, one can find a set of strong generators (SGS) of G in $O(n^4 \log^c n + sn^2)$ time.*

In order to avoid the long expressions as in Theorem 2.1, we introduce a "soft version" of the big-O notation: for two functions $f(n), g(n)$, we write $f(n) = O^{\sim}(g(n))$ if $f(n) \leq Cg(n)\log^c n$ (c, C constants). Thus the time bound in Theorem 2.1 is $O^{\sim}(n^4 + sn^2)$. To appreciate this and later time-bounds, note that s is typically no worse than $O(n^2)$.

It is immediate that

Corollary 2.2. *Given G as above, one can find the order, $|G|$, of G or test membership in G of any permutation in time $O^{\sim}(n^4 + sn^2)$. Additional membership tests cost only $O(n^2)$ each.*

It is sometimes the case, for example in the computation of group characters, that one can make use of a generator-relator presentation of G. It is well known that Sims' SGS representation of permutation groups expedites a generator-relator presentation (see, for example, [Le]), and it is easy to see the translation can be implemented in polynomial-time. It is a corollary to (and a necessary ingredient in) our procedures that we sharpen this observation. We extract the following, which seems of independent interest.

Corollary 2.3. *Given G as above, one can construct in time $O^{\sim}(n^4 + sn^2)$ a generator-relator presentation $\langle X|R\rangle$ of G in which $|X| = O^{\sim}(n)$ and $|R| = O^{\sim}(n^2)$.*

This result is near optimal, for there exist permutation groups G that demand $|X| = \Omega(n)$ and $|R| = \Omega(n^2)$ in any presentation $G = \langle X|R\rangle$.

As we indicated, the Sims-type SGS employs the subgroups that fix, pointwise, initial segments in any ordering of the underlying set. In applications there are reasons, independent of membership-testing, for wanting such subgroups. For example, some graph isomorphism applications require pointwise set-stabilizers (see, e.g., [LM], [Ba86b], [Lu86]). Although our SGS is interpretable in terms of a pointwise stabilizer sequence, some of the points are "new" and, for efficiency, we allow the group structure and action to influence the choice of a sequence. So it is worth mentioning that we do not have to forfeit arbitrary pointwise-set-stabilizers. An obvious (to us) approach is to "sequentialize" the parallel pointwise-set-stabilizer of [BLS]. As it turns out, there is a more direct approach. Brown, Finkelstein, and Purdom [BFP] have described a $O(n^3)$ "base-change" algorithm that gives a direct conversion between SGSs. Thus

Corollary 2.4. *Given G as above and any ordering of the permutation domain, one can construct in time $O^{\sim}(n^4 + sn^2)$ a Sims-type SGS for G with respect to this ordering. In particular, given any subset B of the permutation domain, one can compute generators for the subgroup that fixes the points of B in time $O^{\sim}(n^4 + sn^2)$.*

While an SGS typically occupies $\Omega(n^3)$ space, Jerrum introduced a trick to compress the data-structure to $O(n^2)$ space without loss of efficiency to membership tests [Je]. It is not difficult to impose Jerrum's representation on our procedures so that

Theorem 2.5. *Without loss of time one can require our algorithms to output a Jerrum style compressed ($O(n^2)$ length) description of the SGS constructed. The compressed description supports membership testing in $O(n^2)$ time per test. Furthermore, the workspace required for the construction can be compressed to $O^{\sim}(n^2)$.*

The extent to which substantially more involved algorithms can be improved using the new methods is illustrated by the following result.

Theorem 2.6. *Given a permutation group G by a list of s generators, one can find a composition series of G along with permutation representations of the composition factors in $O^{\sim}(n^4 + sn^2)$ time.*

Here, this is very notable improvement of a previous bound. The problem was shown to be in P in [Lu87] but an implementation of that algorithm could require up to $O(n^8)$ steps.

We discuss the prospects for breaking the $O(n^4)$ barrier on some of these problems. Our procedures reveal some partial information about G in less time. For example,

Theorem 2.7. *Given a permutation group G by a list of s generators, one can list all the primes in $|G|$ in time $O^{\sim}(sn^2)$.*

Essentially, the above follows because we can do the computation for primitive groups within that time bound.

Randomization is used to break two of three principle bottlenecks in the central problem. The case that seemed to be the main obstacle has been the symmetric group. For this we have

Theorem 2.8. *From a given list of generators of the symmetric or alternating group, one can construct an SGS (in Sims' and Jerrum's sense) in $O^{\sim}(n^2 + sn)$ Las Vegas time. (The term "construct" refers to the operations of taking products and powers of permutations.)*

The remaining single n^4 bottleneck is related to the presence of cyclic composition factors. In particular, we derive

Theorem 2.9. *Suppose G has no cyclic composition factors. Then one can find an SGS as well as a composition series of G in $O^{\sim}(\min\{s, n\}n^3)$ Monte Carlo time.*

This illustrates a Monte Carlo vs. Las Vegas bottleneck.

3. Definitions and preliminaries

We refer to any standard text, e.g. [Ha], for basic facts about groups. For permutation group concepts we refer to [Wi] and [Cam]. For information on the classification of finite simple groups ($CFSG$), see [Go], although no knowledge of this work is presumed; the cited consequences have elementary formulations. Cameron [Cam] gives a fine survey of all the consequences of $CFSG$ that are relevant to our work. (We highly recommend that reference as essential source material for work on the complexity of permutation-group computation).

3.1. Group theory

We write $H \leq G$ if H is a subgroup of G. For $S \subset G$, denote by $\langle S \rangle$ the smallest subgroup of G containing S and by $NCl_G(S)$ the *normal closure* of S, that is, the smallest normal subgroup of G containing S.

Let $H \leq G = \langle S \rangle$ and suppose R is a complete set of right coset representatives of G mod H. If $f : G \to R$ is determined by $f(\gamma)\gamma^{-1} \in H$, the set

$$\{f(\rho\sigma)(\rho\sigma)^{-1} : \rho \in R, \sigma \in S\}$$

generates H. These are called *Schreier generators* of H; their number is $|S||G : H|$.

3.2. Permutation groups

The group of all permutations of an m-element set A is denoted $Sym(A)$, or $Sym(m)$ if the specific set is inessential. Subgroups of $Sym(m)$ are the *permutation groups* of *degree* m. The *even* permutations of A form the *alternating group* $Alt(A)$ (or $Alt(m)$). We shall refer to $Sym(A)$ and $Alt(A)$ as the *giants*. These two families of groups require special treatment (see sections 6.1, 8.2).

We say that G *acts on* A if there is a homomorphism $G \to Sym(A)$; such a homomorphism will be specified by indicating the action on A by a set of generators of G. Denote by G^A the image of G under this action; so $G^A \leq Sym(A)$. The *orbit* of $a \in A$ under G is the set of images $\{a^\gamma : \gamma \in G\}$. One says G is *transitive* on A if there is only one orbit, and then G is *t-transitive* if the action of G induced on the set of ordered t-tuples of distinct elements of A is transitive ($t \leq n$). The giants are $(n-2)$-transitive.

Proposition 3.1. *If G is 2-transitive and $|G| \geq n^{2+\log n}$ then G is giant.*

This is an immediate consequence of the classification of doubly transitive groups, which is essentially due to Curtis, Kantor, and Seitz [CKS]. Their work is based on detailed knowledge of the $CFSG$. (For the list of doubly transitive groups, see, e.g., [Cam]).

If G acts on A, the orbits of the induced (componentwise) G-action on $A \times A$ are called *orbitals* [Si76]. The *stabilizer* of $x \in A$ is the subgroup $G_x = \{\gamma \in G : x^\gamma = x\}$. If G is transitive, there is a bijection between the orbitals of G and the orbits of G_x. If G is transitive on A and $G_x = 1$ for some (any) $x \in A$ then G is called *regular*. If G is transitive and $D \subseteq A$, D is called a *block* (for G) if for all $\gamma \in G$, either $D^\gamma = D$ or $D^\gamma \cap D = \emptyset$, and G is called *primitive* if no nontrivial blocks exist. (Trivial blocks have 0, 1 or n elements.) If D is a block then the set of images of D is called a *block system* and an action of G is induced on the block system. The block system is *minimal*, if that action is primitive.

3.3. Primitive groups of Cameron type

An important class of primitive groups is obtained as follows.

First we define a class of imprimitive groups. Let B be a set of $k \geq 5$ elements, $1 \leq s < k/2$. Let $C = rB = B_1 \dot\cup ... \dot\cup B_r$ denote the disjoint union of r copies of B. An s-transversal of C is a subset $X \subset C$ such that $|X \cap B_i| = s$ for $i = 1, ..., r$. Let A denote the set of s-transversals; and $n = |A| = \binom{k}{s}^r$. The *wreath product* $W(B, r) = Sym(B)$ wr $Sym(r) \leq Sym(C)$ consists of all permutations of C that respect the partition $\{B_i\}$. Let now

$$W(B, r) \geq G \geq Alt(B_1) \times ... \times Alt(B_r)$$

and assume G acts transitively on the set of blocks $\{B_i\}$. Under these conditions, the action of G on A is *primitive*. As in [BLS] we term these primitive groups of *Cameron type*.

Proposition 3.2. [Cam] *There exists a constant c such that every primitive group of degree n and order $> n^{c \log n}$ is of Cameron type.*

This, too, is known to be a consequence of the $CFSG$ (proved via a result of Kantor [Ka]). For large n, c approaches 1.

3.4. Strong generators.

In algorithms, permutation groups will always be input and output via a set of generators.

Strong generating sets are a key ingredient in permutation-group computation. An SGS is traditionally defined in terms of a point-stabilizer tower of subgroups (see section 1). We pause to remark on some differences in usage. In the parlance of Sims and others (see, e.g., [Le]), an SGS is any subset of G that includes generators of every G_i. In complexity investigations, it

has been useful to demand that the SGS consist precisely of complete sets of coset representatives as we indicated. (The essential point in any case is that a representative of any coset by quickly constructible from the generators). On the other hand, following [Ba79], it has been customary to generalize in a different direction. Specifically, an SGS is the union of (right) coset representatives C_i in *any* subgroup tower.

$$G = G_0 \geq G_1 \geq \ldots \geq G_r = 1.$$

We always follow the latter point of view, though, except in our second membership algorithm (hinted at in section 10) the $\{G_i\}$ are interpretable as pointwise-stabilizers (albeit in an extended action).

Any $\alpha \in G$ has a unique factorization $\alpha = \rho_r \cdots \rho_2 \rho_1$ with $\rho_i \in C_i$ (and, in any usage, we expect that such factorization is easy to compute, see [FHL]). It is useful to observe that an SGS for a factor group G/N, lifted to G, then appended to an SGS for N, gives an SGS for G. With an abuse of language, we shall call a subset $C \subset G$ a set of (strong) generators of G/N if C is a lifting of such a set. If $\alpha \in G$ is factored according to an SGS of G/N, and $\alpha = \nu\rho_l \cdots \rho_2 \rho_1$ for some $\nu \in N$ then we call ν the *siftee* of α into N. If $C \subset G$ is an SGS for G/N and $S^* \subset G$ is a set of generators for G/N such that $C \subset \langle S^* \rangle$ (i.e., this holds in G, not only in G/N) then we say that S^* is *compatible* with C.

3.5. Sims' algorithm

Our algorithms use an efficient version, described by Knuth [Kn], of Sims' algorithm for computing an SGS. We recall this briefly, and in a convenient setting.

We consider the action of $G = \langle \Phi \rangle \leq Sym(A)$ on the set $C = \{1, 2, \ldots, m\}$. Let G_i be the pointwise stabilizer of $\{1, 2, \ldots, i\}$;

$$G = G_0 \geq G_1 \geq \ldots \geq G_m = N,$$

where N is the kernel of the G-action on C. Our objective is to find an SGS of G/N and normal generators for $N \lhd G$.

During the procedure, we maintain lists T_i, $i = -1, 0, \ldots, m$ and R_i, $i = 0, 1, \ldots, m$, where R_i is a partial list of right coset representatives of G_{i-1} mod G_i; and $T_i \subset G_i$ such that $\langle T_i \rangle \supset \bigcup_{j \geq i+1} R_j$. Set $T_{-1} := \Phi$ and $R_0 := \{1\}$; these will not change during the procedure but other lists may be augmented time to time. Also, we shall maintain the list Ψ of siftees ($\Psi \subset N$).

A subroutine SIFT attempts to factor $\pi \in G_k$ (so the sift starts at the list R_{k+1}) over the current partial coset lists. If it succeeds, it puts the siftee (if not 1) in Ψ; otherwise it inserts a new coset representative in R_j, the point where a representative was not found and adds π to the lists T_i, $k+1 \leq i \leq j-1$.

The main procedure PERMREP applies SIFT to the elements $\rho\tau$ such that (ρ, τ) in $R_{k+1}T_k$ for some k. The trick is always to give preference in exhausting the queue of these (ρ, τ) to those corresponding to the minimal k. As a consequence, we make the critical observation that, each time T_i is augmented, the group $\langle T_i \rangle^C$ increases.

It is not difficult to verify

Proposition 3.3. *When procedure PERMREP terminates, the R_i will be complete sets of coset representatives, and $\langle T_i \rangle^C = G_i^C$. Moreover, $NCl_G(\Psi) = N$.* ♠

To describe the timing, we set $n = |A|$ and assume $n \geq m = |C|$. Therefore, the cost of each group operation is $O(n)$ and a sift costs $O(n \log |G^C|)$. Let $t =$ the size of the final table $= \sum_{i=1}^{m} |G_{i-1}^C : G_i^C|$.

Proposition 3.4. *(a) Let $|\Phi| = M$. The running time of PERMREP is $O(n \log |G^C|(M + t \log |G^C|))$; in particular, if $|G^C| \leq \exp(\log^c n) = \exp(O^\sim(1))$ then the running time is $O^\sim(n(M+m))$.*
(b) At any moment during the execution of the algorithm, $|T_{k-1}| \leq 1 + \sum_{j=k}^{m} \log |R_j|$. ♠

As in [FHL], a slight modification of PERMREP provides an algorithm NORMCL for computing $NCl_G(\Phi)$. We again consider group actions and collect siftees.

The situation is the following: $G = \langle S \rangle \leq Sym(A)$ and $\langle \Phi \rangle \leq Sym(A)$ act on C. The output will be sets of coset representatives $\{R_i\}$ and sets of generators of the stabilizer chain over C for $H := NCl_G(\Phi)$. The set Ψ of siftees collected will be a set of normal generators for the kernel of the H-action on C. In addition to completing the table as above for $\langle \Phi \rangle$, one sifts the elements in T_0^S. Letting $t = \sum_{i=1}^{m} |H_{i-1}^C : H_i^C|$

Proposition 3.5. *Let $M = |\Phi|$. The running time of NORMCL is $O(n \log |H^C|(M + |S| \log |H^C| + t \log |H^C|))$; in particular, if $|H^C| \leq \exp(\log^c n) = \exp(O^\sim(1))$ then the running time is $O^\sim(n(M + |S| + m))$.* ♠

3.6. Structure graphs

A *structure forest* for a permutation group G is a forest on which G acts as automorphisms, fixing the roots, whose leaves constitute the given permutation domain, and such that no non-trivial levels can be inserted that are consistent with the G-action. The latter requirement has the following interpretation: denoting by G_v the subgroups fixing node v and then by $G(v)$ the induced action of G_v on the children of v, we require that $G(v)$ is *primitive*. Structure forests reflect the flow of control in many divide-and-conquer algorithms for permutation groups (treat each root, or orbit, in succession; at a root pass - somehow - to the subgroup that fixes the children, thus restoring intransitivity, etc.) As we indicated, we

need even finer divide-and-conquer. Following [BLS], we describe an *extended structure forest* ($\mathcal{ESF}$) in which we allow "small" trees $T(v)$ to be appended from nodes v of the forest, of course maintaining consistency with the group action. These trees are "small" in that we imagine them to be placed entirely between levels of the initial forest, that is, we fix the nodes of these intermediate levels before descending to the next initial level.

Our particular choice of $T(v)$'s is designed to bring out the natural structure of $G(v)$ when $G(v)$ is a "large group", i.e., a group of Cameron type with order $\geq exp(\log^c n)$; for the present discussion we may take $c = 3$). The insertion of these "small" trees allows us to utilize the structure of $G(v)$ through a different permutation representation.

We use $G1(w)$ to denote the (primitive) permutation group induced by G_w on the set of *immediate children* of the node w of the extended structure forest. ($G1(v) = G(v)$ if no $T(v)$ has been appended at v.) We shall obtain an $\mathcal{ESF}$ such that, at each node, $G1(w)$ is either a small group (i.e. of order $< exp(\log^c n)$) or a giant. In each tree, the groups at a given level are isomorphic; accordingly, we can talk about *small group levels* and *giant levels* in the extended structure forest.

The main algorithm employs a further extension of the $\mathcal{ESF}$, the *structure graph* ($\mathcal{SG}$). At each node w of the $\mathcal{ESF}$ such that $G1(w)$ is a symmetric group, we insert a level containing two points between w and the immediate children of w. let $\tau \in G1(w)$ switch these two new points if and only if τ is an odd permutation, and the action of G is extended consistently. The children of these new points are the immediate children of w in the $\mathcal{ESF}$. For each node w in the $\mathcal{SG}$, $G2(w)$ denotes the permutation group induced by G_w on the set of immediate children of w. ($G2(w) = G1(w)$ if $G1(w)$ is not a symmetric group; $|G2(w)| = 2$ if $G1(w)$ is a symmetric group; $G2(w)$ is an alternating group for the new nodes.) Hence, the $\mathcal{SG}$ consists of small group levels and *alternating levels* only.

4. Organization of the main algorithm

Let $A = A_1 \dot{\cup} A_2 \dot{\cup} ... \dot{\cup} A_r$ be the partition of the permutation domain into orbits, and $F = F_1 \dot{\cup} F_2 \dot{\cup} ... \dot{\cup} F_r$ the extended structure graph where the graph F_i corresponds to the orbit A_i. Let $\mathcal{L}_i$ be the ordered set of levels of F_i (starting from the root), and let $\mathcal{L} = \{L_0, L_1, ..., L_l\}$ be the concatenation of these ordered sets. (We emphasize again that all levels of each appended tree $T(v)$ in the extended structure forest are strictly between two consecutive levels of the corresponding original tree.) Let G^i be the pointwise stabilizer of $L_0 \cup L_1 \cup ... \cup L_i$; so $G^0 \geq G^1 \geq ... \geq G^l = 1$ and G^i is normal in G.

Throughout the algorithm, we shall work with different permutation representations each of degree $O(n)$ of certain subgroups of G. If a procedure performs group operations, we may either need the result in the current representation only (*local* operation) or in the original representation as well (*global* operation). An operation not explicitly labelled "local" will be understood global. Since each permutation will be written in at most three representations, the cost of elementary group operations remains $O(n)$.

Main Procedure

INPUT: a set S of generators for $G \leq Sym(A)$, $|S| = s$.
Step 1. Construct structure forest, and generators of G_v for a representative v of each G-orbit of the nodes.
Step 2. For these representatives, use NATURAL_ACTION to decide whether $G(v)$ is a "large group" and, if so, construct new action and corresponding structure tree $T(v)$.
Step 3. Append a copy of $T(v)$ to all nodes in the orbit v^G, thus obtaining an extended structure forest ($\mathcal{ESF}$). By inserting new levels after giant levels of the $\mathcal{ESF}$, obtain the structure graph ($\mathcal{SG}$). Extend the G-action to the $\mathcal{SG}$; henceforth, compute the effect of any global operation on the entire $\mathcal{SG}$. Compute the node stabilizers G_w as in Step 1 for representatives of G-orbits of the $\mathcal{SG}$.
Step 4. For each node v representing an alternating level in the $\mathcal{SG}$, construct an SGS for $G2(v)$.
Step 5. For $i := 1$ to l do the following:
construct an SGS for G^{i-1}/G^i;
store a compatible generating set S^{i-1} of size $O^\sim(|L_i|)$ for G^{i-1}/G^i;
construct $\Phi^i \subset G^i$ such that $G^i = NCl_G(\Phi^i)$
end.

Using standard procedures for orbits, imprimitivity blocks (e.g. [At]), it is easy to see that Step 1 can be carried out in $O(sn^2)$ time. Step 2 will be discussed in section 5, and we shall conclude that section with the cost of Step 3. We present a novel method for constructing an SGS for the giants in section 6.1. Section 7 relates group presentations to the construction of normal generators. By the results of section 3, the factor groups G^{i-1}/G^i in Step 5 are either subgroups of products of "small" groups, or subgroups of products of alternating groups. We handle the first case in Section 8.1, utilizing NORMCL. For the second case, we use an efficient implementation of Luks' "non-abelian linear algebra" [Lu86] (see section 8.2).

5. Reducing large to giant

The purpose of this section is to organize primitive groups as "large" and "small". Large groups are of Cameron type and we indicate how to recover their "natural" imprimitive action. Thereby most algorith-

mic problems will be reduced to giants and to small groups.

Our objective is achieved by the subroutine NATURAL_ACTION. This procedure is a slight refinement of the one under the same name in [BLS]. The procedure involves a global variable n, the degree of the permutation group which is the input of the full algorithm. We shall always assume that n is sufficiently large.

First, we describe a simple procedure, TEST_GIANT, to test whether or not the (local) action of G on C is a giant, the action already known to be 2-transitive. TEST_GIANT begins to run PERMREP (locally) on C but only while $|\{i : R_i \neq 1\}| < 2\log m + \log^2 m$. If the latter bound is exceeded, it exits and outputs "giant" (recall Proposition 3.1); if PERMREP completes normally, TEST_GIANT outputs "small group". By Proposition 3.4, TEST_GIANT executes in time $O^{\sim}(m^2 + Mm)$, where $|C| = m$.

Procedure NATURAL_ACTION(Φ)

INPUT: a primitive group $G \leq Sym(A)$, where $m := |A| \leq n$.
All group operations executed in this procedure are local.

Step 1. if $m \leq 3\log^2 n$, then (output "small group"; exit).

Step 2. if G is 2-transitive then (TEST_GIANT(G); exit).

Step 3. Consider the orbitals (G-orbits on $A \times A$). Let Γ be the second smallest and Δ the largest orbital. Fix $x \in A$.

if $|\Gamma(x)| > 2\sqrt{m}\log m$ then (output "small group"; exit);

for each $w \in A$ construct $\alpha(w) \in G$ with $x^{\alpha(w)} = w$;

construct Schreier generators for G_x;

fix $y \in \Gamma(x)$. For each $y' \in \Gamma(x)$ compute some $\beta(y') \in G_x$ such that $y^{\beta(y')} = y'$;

compute the sets $B(x,y) = \Delta(y) - \Delta(x)$ and $C(x,y) = A - \bigcup_{z \in B(x,y)} \Delta(z)$;

let $D(x) = \{C(x,y)^{\beta(y')} : y' \in \Gamma(x)\}$;

if $|D(x)| > \log m$ then (output "small group"; exit);

let $D = \{D(x)^{\alpha(w)} : w \in A\}$;

if $|D| > 2\sqrt{m}\log m$ then (output "small group"; exit).

Step 4. Consider G-action on D. (* This action exists and it is transitive. *)

Select a system $\{B_1, ..., B_r\}$ of minimal (nonsingleton) blocks of imprimitivity (* $\bigcup_i B_i = D$ *);

if $k := |B_i| > 4\log n$

and $r \leq \log n/\log 5$

and $G(B_1)$:=the stabilizer of B_1 restricted to B_1 is 2-transitive

and TEST_GIANT($G(B_1)$) returns message "giant"

then (output ("large group, faithfully acting on D", the G-action on D, and a structure tree for the G-action on D in which the nodes next to the leaves represent the blocks B_i); exit)

else (output "small group"; exit).

The justification of NATURAL_ACTION is very close to that of the procedure with the same name in [BLS, section 4]. The main difference stems from the more complex formulation of the set D in Step 3. In [BLS] it was appropriate to compute $\{C(x,\tilde{y}) : (x,y) \in \Gamma\}$ in parallel for all x, y and take this to be D; time constraints prohibit that approach herein. As in [BLS], one shows that if $|G|$ exceeds $\exp(5\log^2 n \log\log n)$ ([BLS] unnecessarily shows "7" instead of "5"), then A can be identified with the s-transversals of $B_1 \dot{\cup} ... \dot{\cup} B_r$, establishing the action as Cameron type. Further, Step 3 of NATURAL_ACTION correctly recovers this structure. (The proof of this part follows as in [BLS] if one adds to Claims 1-5 therein the observation $k < 2\sqrt{m}$; which holds since $rs > 1$ implies $m \geq \binom{k}{2}$). ♠

The timing of these local computations is not difficult. One shows

Proposition 5.2. *Suppose $G = \langle\Phi\rangle$ acts on an m-set, and $|\Phi| = M$. Then NATURAL_ACTION(Φ) runs in $O^{\sim}(Mm^2)$ time.*

Corollary 5.3. *Step 2 of the Main Procedure runs in $O^{\sim}(sn^2)$ time.*

Proof. We apply NATURAL_ACTION to the stabilizers of certain nodes of the structure forest. If $|v^G| = k$ then $mM = O(ks)$, and, in each tree, we call NATURAL_ACTION $O^{\sim}(1)$ times. ♠

We can also now dispose of Step 3 in the Main Procedure. Having computed the pointers identifying pairs in the second smallest orbital with elements of D, the action of any $\sigma \in G_v$ on D, therefore on $T(v)$, can be found in $O^{\sim}(\sqrt{m})$. It follows that an extension of σ to the $S\mathcal{G}$ is obtained in $O^{\sim}(n^2)$. Thus

Proposition 5.4. *Step 3 of the Main Procedure runs in $O^{\sim}(sn^2)$ time.* ♠

6. Managing giants

6.1. Construction of a 3-cycle

Recall that the "giants" are the symmetric and alternating groups in their natural action. We discuss a procedure for *constructing* strong generators of the giants from a given set of generators. Henceforth, we use the term "construction" to mean a sequence of the following *legal operations: multiplication, inversion,* and *taking powers* of permutations.

The reason for the constraint on the set of legal operations is that the procedure is applied to the case when $G2(v)$, for some node v in the extended structure graph, acts on a set B of blocks as an alternating group. In this case, although we know *a priori* that some $\sigma \in G2(v)$ acts on B as, e.g., a given 3-cycle, no such permutation will be guaranteed to belong to G unless it has been constructed, by way of legal operations, from the generators of $G2(v)$.

We refer to [BLS, section 5] for the idea. Based upon the methods of [BS], our procedure, THREE_CYCLE constructs a 3-cycle from the given generators. Once a 3-cycle is available, it is an easy matter to construct an SGS for a giant.

THREE_CYCLE constructs from an element $\tau \in G$ with large *support* (= number of elements actually moved by τ) an element with half the support. There are two main steps, the first involving a carefully chosen commutator $\lambda = \pi^{-1}\tau^{-1}\pi\tau$. λ may have as much as twice the support of τ but it is constructed to have many different primes in its order. The second step exploits this multiplicity of primes to construct a power of λ that cuts the support now by a factor of 4.

A critical ingredient in the above is the ability to construct π with a prescribed action on a set of size $\log^2 n$. Basically, this reduces to constructing complete coset representatives for the first $\log^2 n$ steps in a point-stabilizer tower. PERMREP would require time $\Omega(n^5)$. Instead we work from the top down, constructing Schreier generators for the successive groups $G_1, G_2 \ldots,$. Recall, however, that the number of Schreier generators grows too fast. Thus, we exploit Proposition 3.1, keeping only enough generators around to guarantee 2-transitivity and order $> n^{2+\log n}$. The latter condition is easy to satisfy. For that PERMREP is initially run only until $2\log n + 2\log^2 n$ coset tables are nontrivial (Proposition 3.4(b) controls the number of required sifts). For all successive groups, we shall save the nontrivial entries in the tables numbered above $\log^2 n$. So, from now on, it is only necessary to save enough Schreier generators to guarantee 2-transitivity.

A critical subroutine, then, is PRUNE which cuts down the generating set in a 2-transitive group to a collection of size $< 2n$. The point is that $n-1$ generators will already guarantee transitivity. But then there are at most n orbitals and so $n-1$ additional generators will tie these together.

The timing in THREE_CYCLE is dominated by the calls to PRUNE. In tying the orbitals together, we may have to look at the action of each of the $O(n^2)$ Schreier generators on each of the $O(n^2)$ pairs. This leads to an $O^{\sim}(n^4 + sn^2)$ construction of an SGS. (Note that we typically let "local" consideration guide the construction of a "local" SGS; at least two n's each term are then replaceable by m's).

Remark. We require the computation of $ord(\lambda)$, the order of λ, as well as the powering $\lambda^{ord(\lambda)/p^d}$. Standard multiplication and fast powering methods would suffice in the context of the overall procedure. However, we note that the operations do not require more than $O^{\sim}(n)$ time. The point is that λ cannot have more that $\sqrt{2n}$ different cycle lengths. So $ord(\lambda)$, the least common multiple of these lengths, is computable in $O^{\sim}(n)$ time. It also follows that the number of digits in $ord(\lambda)$ is $O^{\sim}(\sqrt{n})$ (for a more precise estimate, see [La]). But the desired power of λ is computable by reducing the exponent modulo the distinct cycle lengths.

6.2. The Las Vegas SUPERPRUNE

We digress from the Main Procedure to announce a randomized version of THREE_CYCLE with running time $O^{\sim}(sn + n^2)$. The principle speed up is via SUPERPRUNE. The idea is that instead of selecting a subset of the generating set Ψ to ensure 2-transitivity, we take random products of elements of Ψ.

For $g_1, \ldots, g_r$ in G, a *random subproduct* of the g_i is $g_1^{e_1} g_2^{e_2} \cdots g_r^{e_r}$ where the $e_i = 0, 1$ are decided by independent unbiased coin tosses. For any G let $orb(G)$ denote the number of orbits of G.

One sees that, if an orbit O of H is not an orbit of G then, with probability $\geq \frac{1}{2}$, O is not stabilized by a random subproduct of the generators of G. This is used to show

Proposition 6.1 *Suppose $g_1, \ldots, g_r$ generate G and $H \leq G$. Let h be a random subproduct of the g_i. Then the expected value of $orb(\langle H, h\rangle)$ is $\leq orb(G) + \frac{3}{4}(orb(H) - orb(G))$.*

As a consequence, for any fixed d, $O(\log n)$ random products suffice to duplicate the correct orbit structure of G with probability $> 1 - n^{-d}$.

We can apply the above to select $O^{\sim}(1)$ elements in THREE_CYCLE that will certify 2-transitivity. What makes this process Las Vegas in our application is that we can quickly verify the outcome.

The above cuts out one order of magnitude in the process. In fact, still another is removable. The only reason for $O(n^3)$ at this point is that we have set about the task of listing a complete SGS. However, it is easy to see that a *labelled branching* (Jerrum's compact representation of the SGS ensuring factorization of group elements with the same efficiency as complete lists of coset representatives) can be obtained in $O(n^2)$ time.

7. Normal sifting

Recall that by *normal generators* of a normal subgroup, N, of G we mean a set of which the normal closure in

G is N. Let

$$G = G_0 \geq G_1 \geq \ldots G_s = N$$

be a chief series (chain of normal subgroups of G). Let $S_i \subset G_i$ generate G_i mod G_{i+1}, i.e. $G_i = \langle S_i \rangle G_{i+1}$ $(i \leq s-1)$.

We assume a *presentation* $G_i/G_{i+1} = \langle X_i \mid Rel_i \rangle$ is given in which X_i maps bijectively to the images of S_i in G_i. Then for $r_i \in Rel_i$, $r_i(S_i) \in G_{i+1}$.

We also assume that we have an SGS for each G_i/G_{i+1} that is *compatible* with S_i, by which we mean that its elements are actually in $\langle S_i \rangle$ (as opposed to just $\langle S_i \rangle G_{i+1}$). Sifting from G to N consists of sifting through these SGSs successively.

It is not hard to see

Lemma 7.1. (Normal Sift Lemma). *In the above setting, let Q denote the set of the following elements:*

(a) S (the set of generators of G).

(b) $g^{-1}hg$ for $g \in S$ and $h \in S_i$, $1 \leq i \leq s-1$.

(c) $\{r_i(S_i) : r_i \in Rel_i, i \leq s-1\}$.

Then $N = NCl_G(\mathrm{sift}(Q))$.

The Normal Sift Lemma is used each time our breadth-first descent of the structure forest finishes a level. Then G_i will be the pointwise stabilizer of the i^{th} level, L_i. We then work mod G_{i+1}, trying to find normal generators for $N = G_i/G_{i+1}$, the group to be handled in the next round, i.e. till the level L_{i+1} lasts. If this is a small group level, the NORMCL routine (section 3.5) will suffice for finding the required normal closure. On giant levels we have to use an efficient implementation of the *nonabelian linear algebra* described in [Lu86]. The time bounds critically depend on the number of relations to be sifted. Carmichael's set of $< m$ relations defining the alternating group $Alt(m)$ comes in handy ([Car, p.185], or see [CM, p.67]). Not only does this satisfy what is needed for the $O^{\sim}(n^4)$ bound, but, with some additional tricks, it eliminates the giant levels from the list of n^4 bottlenecks.

8. Descending the structure graph

8.1. Small group levels

By the results of section 5, the group $G2(w)$, the action of the stabilizer of the node w in the extended structure graph on the immediate children of w, is either an alternating or a small group. Moreover, for $w, w' \in L_i$ these groups are isomorphic. We call L_i an *alternating level* if $G2(w)$ is alternating for $w \in L_i$, and *small group level* otherwise. Our objective in this subsection is to get past a small group level L_{i-1}. Suppose that we have constructed an SGS for G/G^{i-1}, and normal generators Φ^{i-1} for G^{i-1}, containing the siftees of S into G^{i-1}. Using NORMCL we construct an SGS for G^{i-1}/G^i and normal generators for G^i. We see, via Proposition 3.5 that the running time of this step is $O^{\sim}((|\Phi^{i-1}| + |L_i|s)|L_i|n)$. Furthermore, one observes that $|\Phi^i| \leq |\Phi^{i-1}| + O^{\sim}((s + |L_i|)|L_i|)$.

8.2. Alternating levels

Suppose that we have constructed an SGS for G/G^{i-1} in Step 5 of the Main Procedure. Let $L = L_{i-1}$ be the last level fixed pointwise by G^{i-1}, and suppose that $G2(w)$ is alternating for $w \in L$. We discuss a method to obtain an SGS for G^{i-1}/G^i, and normal generators for G^i.

The nodes in L comprise a G-orbit of the $\mathcal{SG}$; let v be a representative of this orbit. Moreover, let $|L| = p$, and let m be the number of immediate children of v. So G^{i-1} acts on $A = B_1 \dot{\cup} B_2 \dot{\cup} \ldots \dot{\cup} B_r$, ($r = p$ or $r = p/2$); $|B_j| = m$ for all j. Let Φ^{i-1} be the set of normal generators for G^{i-1}, containing the siftees of the original generators into G^{i-1}.

If there exists $\tau \in \Phi^{i-1}$ acting non-trivially on $B_{j'}$ for some $j' \leq r$ then G^{i-1} acts as $Alt(B_{j'})$ on $B_{j'}$; moreover, since L is a G-orbit, $G^{i-1}|_{B_j} = Alt(B_j)$ for all $j \leq r$. Hence, by group-theoretical folklore ([Lu86, Lemma 4.1]; see [Sc, appendix] for a proof), G^{i-1}/G^i is a product of full "diagonal" and so isomorphic to $Alt(m)^k$.

First, we use an efficient procedure to compute an SGS for an alternating group as the normal closure of a non-trivial permutation π (NORMCL, as indicated in section 3.5, is too slow for present purposes). The trick is again to construct a 3-cycle. However, the setting is somewhat more tractable than that for THREE_CYCLE for we are able to conjugate by members of the full alternating group. In fact, a commutator of π with an appropriate 3-cycle yields a non-trivial element that moves at most 6-points. Another commutator by an element of the full group yields a 3-cycle in the normal closure.

Having collected enough elements to guarantee a full alternating action in each coordinate, we necessarily are dealing with a product of diagonal actions, that is, collections of actions are "linked". However, at this point, there may still be links that do not persist in the full normal closure. The main part of the procedure involves an efficient implementation of Luks' "non-abelian linear algebra" to determine which coordinates of $\prod_{j \leq r} Alt(B_j)$ must now be unlinked. Note that as long as we have not achieved normal closure there is a generator of the full group that can break a link. One also uses the fact that the full group action is transitive on the coordinates, so that images of a smallest linked collection break any larger ones; if another link can be broken at this point, the smallest size will have been halved.

Having established the quotient G_i/G_{i+1} as a product of alternating groups, a presentation follows easily from

a short presentation of the alternating group. We note that we have to include relations that certify each pair of factors commute. Since these relations will later have to be sifted, and a sift costs $O(n^2)$, this fact seems to contribute to the $O(n^4)$ term in the time bound. However, this particular bottleneck is removable, for it actually suffices to guarantee that the first alternating group commutes with the others, for the transitive action of the overall group carries the contribution of these relations to the others (as we take normal closures).

9. Composition factors

Because of the overhead in establishing the notation and background of the composition factors algorithm, we omit most of this discussion for this Proceedings. However, a few comments are useful to establish the scenario for one of the $O(n^4)$ bottlenecks.

The giant levels have been handled. We need only find composition factors for the small group levels, in other words, where a small group is induced in each orbit.

The problem reduces to just the small group case. For, if one has cracked the problem for small groups, one can *augment* the $\mathcal{SG}$ with a chief series at each node (much as the structure forests are augmented in [Lu86]), and descend the augmented graph via normal sifting. This time we are faced with levels in which the group is a direct product of isomorphic simple groups. Linear algebra or "nonabelian linear algebra" prevails.

So assume G is a small group, i.e., $|G| \leq n^{c \log n}$. By section 3.4, the fundamental operations (e.g., membership, normal closure) can be done in time $O^{\sim}(n^2)$.

The composition factors procedure closely follows [Lu86] which is now our essential reference. The problem is easily reduced to the primitive case. Then, a number of types of tests are made, mostly seeking kernels of induced actions. Either G is simple or one of these reveals a proper normal subgroup. We can speedup or modify all but one of these test types, while also limiting the number of calls to each.

There is just one holdout. It is the test designed to bring out a normal subgroup when G has an abelian socle. (Recall that the *socle* is generated by all minimal subgroups; if the socle of a primitive group is abelian, it is, perforce, regular). Our best algorithm is as follows: fix a point x; choose y in a smallest orbit of G_x; let $H :=$ the normalizer of G_{xy} in G (tack on to G_{xy} those coset representatives that normalize it); now try to embed H in a maximal subgroup M of G; find the kernel of the action of G on the cosets of M (the action cannot be faithful since the socle is not regular; H contains the socle element that maps x to y).

Embedding H in a maximal subgroup is equivalent to finding a minimal block system in the action of G on the cosets of H. But the problem is that the number of points in the latter action is $O(n^2)$, whence the present $O(n^4)$ deterministic timing.

But, one can seek M by choosing random cosets of H, attaching these to H. We need only find one that reduces the index in G. If so, we repeat as long as the index remains $> n$, for we know (if G has an abelian socle) that our group is not yet maximal. We need only guess $O^{\sim}(1)$ cosets to have high probability of success. (again, if G has an abelian socle). Note, though, that failure is inconclusive for G may also have been simple.

10. The bottlenecks

As we indicated, the composition factors algorithm requires an augmentation of the $\mathcal{SG}$ so that levels involve manageable products of simple groups. But a corollary to the process is a reformulation of the basic algorithm for membership, for we can form a (generalized) SGS along the way. The timing of the new algorithm can be expressed as $O^{\sim}(n^4 + sn^2 + \min\{s, n\}n^3)$. However, (assuming small s) this method comes close to breaking the n^4 barrier. Indeed, there are now three, well-separated, n^4 bottlenecks, while the other routines can be executed in $O^{\sim}(\min\{s, n\}n^3)$ time.

Two of the bottlenecks have been mentioned herein as we indicated how randomization helps.

Bottleneck I occurs in the problem of pruning $O(n^2)$ generators of an alternating group to size $O(n)$. PRUNE does this in $O^{\sim}(n^4)$ time. The Las Vegas SUPER-PRUNE speeds this up to $O^{\sim}(n^3)$. (See section 6).

Bottleneck II occurs in dealing with primitive groups with abelian socle. Deterministically, this case is requiring $O^{\sim}(n^4)$ time. A Monte-Carlo version runs in $O^{\sim}(n^3)$. (Section 9).

Bottleneck III stems from the number of relations in an abelian level of the *augmented* $\mathcal{SG}$. For nonabelian levels, the number of relations can be controlled by the trick mentioned at the end of section 8.2. However, in the abelian case, we are, as yet, unable to cut down on the $O(n^2)$ relations that will have to be sifted at a cost of $O(n^2)$ each. The irony is the siftees are only used as normal generators for the group farther down; a permutation group cannot *need* $O(n^2)$ generators.

References

[At] M.D. Atkinson, *An algorithm for finding the blocks of a permutation group*, Math. Comp. 29 (1975), 911–913.

[Ba79] L. Babai, *Monte Carlo algorithms for graph isomorphism testing*, Tech. Rep. 79–10, Universite de Montreal 1979.

[Ba83] L. Babai, "Permutation Groups, Coherent Configurations, and Graph Isomorphism," D. Sc. Thesis, Hungarian Academy of Sciences 1983 (in Hungarian).

[Ba86a] L. Babai, *On the length of subgroup chains in the symmetric group*, Communications in Algebra 14 (1986), 1729–1736.

[Ba86b] L. Babai, *A Las Vegas-NC algorithm for isomorphism of graphs with bounded multiplicity of eigenvalues*, Proc. 27th IEEE FOCS, Toronto 1986, pp. 303–312.

[BFP] C.A. Brown, L. Finkelstein, and P.W. Purdom, *A new base change algorithm for permutation groups*, to appear.

[BKL] L. Babai, W.M. Kantor and E.M. Luks, *Computational complexity and the classification of finite simple groups*, Proc. 24th IEEE FOCS, 1983, pp. 162–171.

[BL] L. Babai and E.M. Luks, *Canonical labeling of graphs*, Proc. 15th ACM STOC, 1983, 171–183.

[BLS] L. Babai, E.M. Luks, and Á. Seress, *Permutation groups in NC*, Proc. 19th ACM STOC, 1987, 409–420.

[BS] L. Babai and Á. Seress, *On the degree of transitivity of permutation groups: a short proof*, J. Combinatorial Theory (A) 45(1987), 310–315.

[Cam] P.J. Cameron, *Finite permutation groups and finite simple groups*, Bull. London Math. Soc. 13 (1981), 1–22.

[Car] R. Carmichael, "Introduction to the Theory of Groups of Finite Order," Dover, New York 1956.

[CKS] C.W. Curtis, W.M. Kantor and G.L. Seitz, *The 2-transitive permutation representations of the finite Chevalley groups*, Trans. A.M.S. 218 (1976), 1–57.

[CM] H.S.M. Coxeter and W.O.J. Moser, "Generators and Relations for Discrete Groups," Third Ed., Springer 1972.

[FHL] M.L. Furst, J. Hopcroft and E.M. Luks, *Polynomial time algorithms for permutation groups*, Proc. 21th IEEE FOCS, 1980, pp. 36–41.

[GHLSW] Z. Galil, C.M. Hoffmann, E.M. Luks, C.P. Schnorr, A. Weber, *An $O(n^3 \log n)$ deterministic and an $O(n^3)$ Las Vegas Isomorphism test for trivalent graphs*, Journal of the ACM, 34 (1987) 513–531.

[Go] D. Gorenstein, "Finite Simple Groups and Their Classification," Academic Press, 1986

[Ha] M. Hall, Jr., "The Theory of Groups", Macmillan, N.Y. 1959.

[Ho] C.M. Hoffmann, "Group Theoretic Algorithms and Graph Isomorphism," Lect. Notes in Comp. Sci. 136, Springer 1982.

[Je82] M. Jerrum, *A compact representation for permutation groups*, Proc. 23rd IEEE FOCS, 1982, 126–133.

[Je86] M. Jerrum, *A compact representation for permutation groups*, J. of Algorithms 7 (1986), 60–78.

[Ka] W.M. Kantor, *Permutation representations of the finite classical groups of small degree or rank*, J. Algebra 60 (1979), 158–168.

[Kn] D.E. Knuth, *Notes on efficient representation of perm groups*, unpublished notes, 1981.

[Ko] D. Kozen, *Lower bounds for natural proof systems*, Proc. 18th IEEE FOCS, 1977, 254–266.

[La] E. Landau, "Handbuch der Lehre von der Verteilung der Primzahlen," Teubner, Leipzig, 1909. Reprinted, Chelsea, New York, 1953.

[Le] J.S. Leon, *On an algorithm for finding a base and strong generating set for a group given by generating permutations*, Math. Comp. 35, 941–974.

[Lu82] E.M. Luks, *Isomorphism of graphs of bounded valence can be tested in polynomial time*, J. Comp. Sys. Sci. 25 (1982), 42–65.

[Lu86] E.M. Luks, *Parallel algorithms for permutation groups and graph isomorphism*, Proc. 27th IEEE Symp. on Foundations of Computer Science, Toronto 1986, pp. 292–302.

[Lu87] E.M. Luks, *Computing the composition factors of a permutation group in polynomial time*, Combinatorica 7 (1987), 87–99.

[LM] E.M. Luks and P. McKenzie, *Fast parallel computation with permutation groups*, Proc. 26th IEEE FOCS, 1985, pp. 505–514.

[Re] J.H. Reif, *Probabilistic algorithms in group theory*, TR-01-85, Aiken Computation Lab, Harvard.

[Sc] L.L. Scott, *Representations in characteristic p*, in: Proc. Santa Cruz Conf. on Finite Groups, A.M.S. 1980, pp. 319–322.

[Si67] C.C. Sims, *Graphs and finite permutation groups*, Math. Z. 95 (1967), 76–86.

[Si70] C.C. Sims, *Computational methods in the study of permutation groups*, in: "Computational Problems in Abstract Algebra," J. Leech, ed., Pergamon Press 1970, pp. 169–183.

[Wi] H. Wielandt, "Finite Permutation Groups," Academic Press, N.Y. 1964.