PERMUTATION GROUP ALGORITHMS VIA BLACK BOX RECOGNITION ALGORITHMS

WILLIAM M. KANTOR* and ÁKOS SERESS†

*University of Oregon, Eugene, OR 97403, U.S.A.
†The Ohio State University, Columbus, OH 43210, U.S.A.

Abstract

If a black box simple group is known to be isomorphic to a classical group over a field of known characteristic, a Las Vegas algorithm is used to produce an explicit isomorphism. This is used to upgrade all nearly linear time Monte Carlo permutation group algorithms to Las Vegas algorithms when the input group has no composition factor isomorphic to an exceptional group of Lie type or a 3–dimensional unitary group.

Key words and phrases: computational group theory, black box groups, classical groups, matrix group recognition
1991 Mathematics Subject Classification: Primary 20B40, 20G40; Secondary: 20P05, 68Q25, 68Q40

1 Introduction

There is a large library of nearly linear time permutation group algorithms [BCFS, BS, CF, LS, Mo, Ra, SchS, Ser]. Most of these are Monte Carlo (which means that the algorithm can return an incorrect answer, although the probability of that can be made as small as desired). The main result of this note is that Monte Carlo can be upgraded to Las Vegas (which means that the output is always correct, but the algorithm may also report failure, although the probability of that can be made as small as desired), whenever there are suitable recognition algorithms for the simple groups occurring as composition factors.

There is a growing literature of recognition algorithms for quasisimple groups of Lie type. The first of these, due to Neumann and Praeger [NP], solved the following problem: given a group $G \leq \mathrm{GL}(d,q)$ by a set of generating matrices, decide whether G contains $\mathrm{SL}(d,q)$. Subsequently, this result was generalized to the other classical groups [NiP1, NiP2, CLG1, CLG2, Ce, Pr], also assuming that matrices were given of the desired size over the desired field. A much more general setting is where a quasisimple matrix group is given only as a black box group. Recently, Cooperman, Finkelstein and Linton [CFL] studied the case $G \cong \mathrm{PSL}(d,2)$, providing a methodology for handling many such questions simultaneously. We extended their result in [KS] to all classical groups over all finite fields. In [BCFL] black box groups isomorphic to $\mathrm{PSL}(d,q)$ will also be dealt with for any q in a manner similar to [CFL].

A *black box group* G is a group whose elements are encoded as 0-1 strings of uniform length N, and the group operations are performed by an oracle (the "black

box"). Given strings representing $g, h \in G$, the black box can compute strings representing gh and g^{-1}, and decide whether or not $g = h$. Note that $|G| \leq 2^N$: we have an upper bound on $|G|$. Algorithms usually try to exploit the specific features of the representation of the group they work with. By contrast, a black box group algorithm does not rely on specific features of the group representation or on particulars of how the group operations are performed. It turns out that this is a critical aspect of our uses for these algorithms (cf. Section 2.3).

We state our results about classical groups in a more general setting so their applications for permutation groups can be extended easily when recognition algorithms for additional groups become available.

Definition 1.1 Let $\mathcal{F}$ be a family of simple groups and $f: \mathcal{F} \to \mathbb{R}$ a function taking positive values. We say that $\mathcal{F}$ is *black box f-recognizable* if, whenever a group $G = \langle \mathcal{S} \rangle$ isomorphic to a member of $\mathcal{F}$ is given as a black box group encoded by strings of length N and, in the case of Lie-type G, the characteristic of G is given, there are Las Vegas algorithms for the following:

(i) Find the isomorphism type of G.

(ii) Find a new set $\mathcal{S}^*$ of size $O(N)$ generating G, and a presentation of length $O(N^2)$ in terms of $\mathcal{S}^*$. (This presentation proves that G has the isomorphism type determined in (i).)

(iii) Given $g \in G$, find a straight-line program of length $O(N)$ from $\mathcal{S}^*$ to g.

Moreover,

(iv) The algorithms for (i)–(iii) run in time $O((\xi + \mu) f(G) N^c)$, where ξ is an upper bound on the time requirement per element for the construction of independent, (nearly) uniformly distributed random elements of G, μ is an upper bound on the time required for each group operation in G, and c is an absolute constant.

A *straight-line program of length m* reaching some $g \in G$ can be thought of as a sequence of group elements $(g_1, \ldots, g_m)$ such that $g_m = g$ and for each i one of the following holds: $g_i \in \mathcal{S}^*$, or $g_i = g_j^{-1}$ for some $j < i$, or $g_i = g_j g_k$ for some $j, k < i$. More precisely, since we do not want to store the group elements themselves, the straight-line program reaching g is a sequence of expressions $(w_1, \ldots, w_m)$ such that, for each i, either w_i is a symbol for some element of $\mathcal{S}^*$, or $w_i = (w_j, -1)$ for some $j < i$, or $w_i = (w_j, w_k)$ for some $j, k < i$, such that if the expressions are evaluated the obvious way then the value of w_m is g. This more abstract definition not only requires less memory, but also *enables us to construct a straight-line program in one representation of G and evaluate it in another*, which is an important feature of the algorithms.

We shall prove the following two theorems. Let $\mathcal{G}$ denote the family of all finite simple groups, and let $m: \mathcal{G} \to \mathbb{R}$ be the function such that $m(G)$ is the degree of the smallest faithful permutation representation of G.

Theorem 1.2 *Given a permutation group $G = \langle \mathcal{S} \rangle \leq S_n$ such that all nonabelian composition factors of G are from a black box m-recognizable family $\mathcal{F}$, a base and*

strong generating set for G can be computed in nearly linear Las Vegas time.

Bases and strong generating sets (SGS) are the basic data structures in algorithms for permutation groups; we shall define them in Section 2.1. We call an algorithm *nearly linear* if its running time is of the form $O(n|S|\log^k|G|)$ for some constant k. We shall justify the name and elaborate more on this notion in Section 2.2.

The novelty in Theorem 1.2 is that the base and SGS construction is Las Vegas. Earlier nearly linear time algorithms used the Monte Carlo construction of [BCFS]. All currently known nearly linear Monte Carlo algorithms can be modified so that after an initial base and SGS computation, all further steps of the algorithm are deterministic or Las Vegas. Thus, *for the groups described in* Theorem 1.2, *we can upgrade the entire nearly linear time library to Las Vegas.*

The algorithm in Theorem 1.2 differs significantly from the traditional SGS constructions [Si1, Si2]; by the time we have found $|G|$ we have also constructed a composition series for G. In this respect, the algorithm resembles the parallel handling of permutation groups [BLS1] and the current fastest deterministic algorithms for computing strong generating sets [BLS2, BLS3].

The second theorem is a constructive version of a result from [BGKLP] about short presentations of groups.

Theorem 1.3 *There is a nearly linear Las Vegas algorithm which, when given a permutation group G satisfying the composition factor restriction of* Theorem 1.2, *computes a presentation of length $O(\log^3|G|)$ for G.*

Using the terminology of this paper, the main result of [KS] can be stated as:

Theorem 1.4 [KS] *The classical simple groups, with the possible exception of the 3-dimensional unitary groups, comprise a black box m-recognizable family.*

We shall also use a similar result for the alternating groups.

Theorem 1.5 [BLNPS] *The alternating groups comprise a 1-recognizable family (i.e., one can take $f(G) = 1$ for all alternating groups G).*

It is easy to check that cyclic groups of prime order are m-recognizable and, obviously, sporadic simple groups are 1-recognizable. Hence, combining the previous two theorems with Theorem 1.2 we obtain the

Corollary 1.6 *Given a permutation group $G \leq S_n$ with no exceptional Lie type or 3-dimensional unitary composition factors, all known nearly linear time algorithms dealing with G can be upgraded to Las Vegas algorithms.*

We note that in [KS] a new generating set $\mathcal{S}^*$ satisfying Definition 1.1(iii) was found within the required time bound in 3-dimensional unitary groups as well, but it is an open problem whether these groups have a presentation of length $O(\log^2|G|)$ as needed in 1.1(ii).

Actually, in [KS] we prove more than Theorem 1.4: an isomorphism λ with a group of matrices in the correct dimension is constructed, defined by the images of generators, together with procedures to compute the image of any element of G under λ or of any element of $G\lambda$ under λ^{-1}. The analogous isomorphism for alternating groups is constructed in [BLNPS]. These procedures are very useful for further computations with G, such as the construction of Sylow subgroups [Ka, Mo]. For possible applications of [KS] in matrix recognition algorithms see [BB] and [Pr] in these Proceedings.

In [KS] there are also more precise timings of algorithms than required in Theorem 1.4. For example, we show that the family of classical groups, with the possible exception of 3-dimensional unitary groups, is black box f-recognizable for the function

$$f(G) = \begin{cases} q & \text{if } G \cong \mathrm{PSL}(d,q) \text{ for some } d \\ q^2 & \text{for all other } G \text{ defined on a vector space over } \mathrm{GF}(q). \end{cases}$$

It seems very likely that the set of *all* groups of Lie type is a black box f-recognizable family with $f(G) \leq m(G)$. Research is presently under way on the groups of Lie rank ≥ 2 other than ${}^2F_4(q)$. Possibly the biggest obstacle is condition (ii) of Definition 1.1 in the case of rank 1 groups: finding $O(\log^c |G|)$-length presentations for $\mathrm{PSU}(3,q)$, ${}^2B_2(q)$ and ${}^2G_2(q)$ has been a very annoying open problem for several years (cf. [BGKLP]).

2 The proofs

2.1 Bases, strong generating sets, and Schreier trees

Fundamental data structures for computing with permutation groups were introduced by Sims in [Si1, Si2]. A *base* for a permutation group $G \leq \mathrm{Sym}(\Omega)$ of degree n is a sequence $B = (\beta_1, \ldots, \beta_M)$ of points from Ω such that the pointwise stabilizer $G_B = 1$. The *point-stabilizer chain* of G relative to B is the chain of subgroups

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

where $G^{(i)} = G_{(\beta_1,\ldots,\beta_{i-1})}$. The base B is called *nonredundant* if there is strict inclusion $G^{(i)} > G^{(i+1)}$ for all $1 \leq i \leq M$; then $(\log|G|)/(\log n) \leq |B| \leq \log|G|$. A *strong generating set* (SGS) for G relative to B is a set S of generators of G with the property that

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

Let $B = (\beta_1, \ldots, \beta_M)$ be a base of the group G, let $G = G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(M+1)} = 1$ be the corresponding point-stabilizer chain, and let R_i denote a transversal for $G^{(i+1)}$ in $G^{(i)}$, $1 \leq i \leq M$. Such a transversal can be computed from the SGS by a standard orbit computation of $\beta_i^{G^{(i)}}$, keeping track of the group elements sending β_i to the points of the orbit. Each $g \in G$ can be written uniquely in the form

$$g = r_M r_{M-1} \cdots r_2 r_1, \; r_i \in R_i. \tag{2.1}$$

The process of factoring g in this form is called *sifting* or *stripping*. Note that the order of G can be obtained easily as $|G| = \prod_{i=1}^{M} |R_i|$.

In practical computation, the transversals R_i usually are not computed and stored explicitly; rather, they are encoded in a Schreier-tree data structure. Suppose that a base B and an SGS $\mathcal{S}$ for G relative to B are given. A *Schreier-tree data structure* for G is a sequence of pairs (S_i, T_i) called *Schreier trees*, one for each base point β_i, $1 \leq i \leq M$, where T_i is a directed labeled tree with all edges directed toward the root β_i, and with edge-labels selected from the set $S_i := \mathcal{S} \cap G^{(i)} \subseteq G^{(i)}$. The nodes of T_i are the points of the orbit $\beta_i^{G^{(i)}}$. The labels satisfy the condition that, for each directed edge from γ to δ with label h, $\gamma^h = \delta$. If γ is a node of T_i, then the sequence of the edge-labels along the path from γ to β_i in T_i is a word in the elements of S_i such that the product of these permutations moves γ to β_i. Thus each Schreier tree (S_i, T_i) defines *inverses* of the elements of the transversal R_i for $G^{(i+1)}$ in $G^{(i)}$.

Given an arbitrary SGS $\mathcal{S}$ relative to B, an algorithm in [BCFS] constructs a new SGS $\mathcal{T}$ in $O(nM|\mathcal{S}|\log^2|G|)$ deterministic time such that the depth of each Schreier tree defined by $\mathcal{T}$ is at most $2\log|G|$. We call a Schreier-tree data structure *shallow* if the depth of each tree is at most $2\log|G|$. A shallow Schreier-tree data structure supports membership testing in $O(nM\log|G|)$ time. We will assume that all bases computed in our algorithms are nonredundant and that all Schreier-tree data structures we consider are shallow.

2.2 Nearly linear time algorithms

In groups of current interest for implementations, it frequently happens that the degree of $G = \langle S \rangle$ is in the tens of thousands or even higher, so even a $\Theta(n^2)$ algorithm may not be practical. On the other hand, $\log|G|$ is often small. Therefore, a recent trend is to search for algorithms with running time of the form $O(n|\mathcal{S}|\log^k|G|)$. More precisely, given a constant c, a family $\mathcal{G}$ of permutation groups is called a family of *small-base groups* if all $G \in \mathcal{G}$ of degree n admit bases of size $O(\log^c n)$; or, equivalently, if there is a constant c' such that $\log|G| = O(\log^{c'} n)$ for each $G \in \mathcal{G}$ of degree n. For example, all classical simple groups, in all of their permutation representations, comprise a small-base family (with $c = 2$).

We call a permutation group algorithm a *nearly linear time algorithm* if its running time for any $G = \langle S \rangle \leq S_n$ is $O(n|S|\log^k|G|)$. The name is justified by the fact that, if G is a member of a small-base family then the running time is a nearly linear, $O(n|S|\log^{c''}(n|S|))$, function of the input length. We will require the following algorithms of this sort:

Theorem 2.2 *There are Monte Carlo nearly linear time algorithms which, when given $G \leq S_n$, find the following:*

(i) [BCFS] *A base, strong generating set, and a shallow Schreier-tree data structure for G;*

(ii) *As a consequence of* (i): *given a homomorphism $\varphi: G \to S_n$ specified by the images of generators, data structures which enable the nearly linear time*

computation of $\varphi(g)$ for any $g \in G$ and a preimage of any $g \in \varphi(G)$;

(iii) [BS] *A composition series $G = N_1 \rhd N_2 \rhd \cdots \rhd N_m = 1$ and, for each $1 \le i \le m-1$, a homomorphism $\varphi_i: N_i \to S_n$ with $\ker \varphi_i = N_{i+1}$.*

A large part of the permutation group library in GAP [Sch+] consists of implementations of nearly linear algorithms.

2.3 Permutation groups as black box groups

Suppose that a base $B = (\beta_1, \ldots, \beta_M)$, a strong generating set $\mathcal{S}$ with respect to B, and a shallow Schreier-tree data structure $\mathcal{ST} = \{(S_i, T_i) \mid 1 \le i \le M\}$ are given for some $G \le \mathrm{Sym}(\Omega)$, where the sum of the depths of these M Schreier trees is $t = O(\log^2 |G|)$. We may assume that the SGS $\mathcal{S}$ is closed under taking inverses.

Any $g \in G$ can be written uniquely in the form (2.1) for elements r_i of the transversals whose inverses were coded by $\mathcal{ST}$. Each such inverse can be written as a word in the strong generators $\mathcal{S}$, following the path in the appropriate Schreier tree. Taking the inverse of this word and using the fact that $\mathcal{S} = \mathcal{S}^{-1}$, we obtain the r_i, and so g, as a word in $\mathcal{S}$ in a well-defined way. The length of the word representing g is at most t; this word is called the *standard word representing* g. We note the following:

Lemma 2.3 *In deterministic $O(t|B|)$ time, given an injection $f: B \to \Omega$, it is possible to find a standard word representing some $g \in G$ with $B^g = f(B)$ or to determine that no such element of G exists.*

This algorithm *relies on a base, SGS and Schreier-tree data structure.* However, those inputs are computed by Monte Carlo algorithms, and hence may not be correct. Therefore, it is possible that the preceding algorithm returns an incorrect answer — though with small probability.

Now *we show how to consider G as a black box group H*; this will be crucial for Theorems 1.2 and 1.3. The elements of H are defined to be the standard words representing the elements of G; these are strings over the alphabet $\mathcal{S}$, of length at most t. Of course, we can write the elements of H as 0-1 strings of uniform length N: $\mathcal{S}$ can be coded by $\lceil \log(|\mathcal{S}|+1) \rceil$–long 0-1 sequences for the numbers $1, 2, \ldots, |\mathcal{S}|$; every standard word can be padded by 0's to length t. Since $|\mathcal{S}|$ and t are $O(\log^2 |G|)$, N is $O(\log^2 |G| \log\log |G|)$. As customary in the analysis of permutation group algorithms, we assume that small numbers can be read in $O(1)$ time, and therefore we shall ignore the factor $\log\log |G|$ above; this is at worst $\log n$, and hence is appropriate within the nearly linear time context.

Formally, we have an isomorphism $\psi: G \to H$ with the following properties. Each $g \in G$ defines an injection $f: B \to \Omega$ by $f(\beta_i) := \beta_i^g$, and then Lemma 2.3 can be used to compute $g\psi$ in $O(t|B|) = O(\log^3 |G|)$ time. Conversely, given $h \in H$, $h\psi^{-1}$ can be computed in $O(n \log^2 |G|)$ time, by multiplying out the product of the elements of h as a permutation.

Each $h \in H$ is represented by a unique string, so comparison of group elements can be performed in $O(\log^2 |G|)$ time. In order to take the product of $h_1, h_2 \in H$,

we concatenate these two words, and define a function $f: B \to \Omega$ by $f(\beta_i) := \beta_i^{h_1 h_2}$. Then the standard word representing $h_1 h_2$ can be obtained by Lemma 2.3. This procedure runs in $O(\log^3 |G|)$ time. Similarly, to take the inverse of some $h \in H$, we take the inverse of the word h. This defines an injection $f: B \to \Omega$, and again we use Lemma 2.3 in $O(\log^3 |G|)$ time. Hence, *we have a black box oracle which performs the black box group operations in* $O(\log^3 |G|)$ *time*, which is potentially faster than the ordinary permutation multiplication. In particular, if $G \leq S_n$ is a member of a small-base family, then, in the notation of Definition 1.1, $\mu = O(\log^c n)$ for some constant c. Recall, however, that this oracle can give incorrect answers if our base, SGS or Schreier-tree data structure was incorrect.

(Nearly) random elements of $G\psi$ and of subgroups of $G\psi$ can be constructed, by a remarkable algorithm of Babai [Ba], in $O(\mu \log^5 |G|)$ time. (An apparently practical heuristic algorithm for this purpose is given in [CLMNO].)

Summarizing, we can construct an isomorphism between a permutation group $G \leq S_n$ and a black box group H such that the word length N of the encoding of H, as well as the time requirement for the group operations in H and constructing random elements in H are bounded from above by a polylogarithmic function of $\log |G|$. Therefore, we can perform $O(n)$ group operations in H within nearly linear time. Note that if we considered G as a black box group, with the original permutation multiplication as black box group operation, then $O(n)$ group operations would result in an $O(n^2)$ algorithm.

2.4 Proofs of Theorems 1.2 and 1.3

Proof of Theorem 1.2 Let $G = \langle T \rangle \leq S_n$ be a permutation group. Compute a base and strong generating set, and a composition series $G = N_1 \rhd N_2 \rhd \cdots \rhd N_m = 1$, by the nearly linear Monte Carlo algorithms in Theorem 2.2. The composition series algorithm also provides homomorphisms $\varphi_i: N_i \to S_n$ with $\ker \varphi_i = N_{i+1}$, for $1 \leq i \leq m-1$. We also compute strong generating sets for all N_i with respect to the base of G. We will verify the correctness of the base and strong generating sets for the subgroups N_i by induction on $i = m, m-1, \ldots, 1$.

Suppose that we already have verified an SGS for N_{i+1}. Using Theorem 2.2, we compute a base, SGS, shallow Schreier-tree data structure, and an isomorphism ψ_i with a black box group for the image $N_i\varphi_i$ (cf. Section 2.3), which is a subgroup of S_n and is allegedly isomorphic to a simple group. Our first goal is to *obtain in nearly linear Las Vegas time a presentation of length* $O(\log^2 |N_i\varphi_i|)$ *for* $N_i\varphi_i$, using a generating set $\mathcal{S}_i^*$ such that a straight-line program of length $O(\log |N_i\varphi_i|)$ from $\mathcal{S}_i^*$ to any given element of $N_i\varphi_i$ can be obtained in nearly linear Las Vegas time.

As a consequence of the classification of finite simple groups, we know that there are no three pairwise nonisomorphic simple groups of the same order. So we have at most two candidate simple groups C for the isomorphism type of $N_i\varphi_i$, and in the ambiguous cases we try both possibilities. Also, if $|N_i\varphi_i| > 8!/2$ then $|N_i\varphi_i|$ determines whether $N_i\varphi_i$ is of Lie type, and if it is, its characteristic. Hence, using that $N_i\varphi_i$ is from an m-recognizable family, we can obtain $\mathcal{S}_i^*\psi_i$, a presentation, and straight-line programs in $N_i\varphi_i\psi_i$, in nearly linear Las Vegas time. By Theorem 2.2(ii), the preimage $\mathcal{S}_i^*$ of $\mathcal{S}_i^*\psi_i$ can also be obtained in nearly linear time.

Now the correctness of the SGS for N_i can be proved the following way. Let $\mathcal{T}_i$ be the set of generators of N_i computed by the composition series algorithm (Theorem 2.2(iii)). We check that (i) $N_{i+1} \lhd N_i$ and $N_i \neq N_{i+1}$; (ii) $\langle \mathcal{S}_i^* \varphi_i^{-1} \rangle N_{i+1}/N_{i+1}$ satisfies the presentation computed for $N_i\varphi_i$; and (iii) $\mathcal{T}_i \subset \langle \mathcal{S}_i^* \varphi_i^{-1} \rangle N_{i+1}$, where $h\varphi_i^{-1}$ denotes a lift (i.e., an arbitrary preimage) of $h \in \mathcal{S}_i^* \subset N_i\varphi_i$. Checking (i)–(iii) shows that $|N_i| = |N_{i+1}||N_i\varphi_i|$. If $|N_i|$ is equal with the value for $|N_i|$ computed from the alleged SGS of N_i then the SGS construction is correct: it is known that the alleged order of a group obtained from the Monte Carlo SGS construction is not greater than the true order, with equality if and only if the SGS construction is correct.

For (i), conjugate the generators of N_{i+1} by the elements of $\mathcal{T}_i$, and check that the resulting permutations are in N_{i+1} (since the correctness of N_{i+1} is already known, membership testing giving guaranteed correct results is available for that group). Also, check that not all elements of $\mathcal{T}_i$ are in N_{i+1}. For (ii), multiply out the relators that were written in terms of $\mathcal{S}_i^*$, using the permutations in $\mathcal{S}_i^* \varphi_i^{-1}$; then check that the resulting permutations are in N_{i+1}. Finally, for (iii) write straight-line programs from $\mathcal{S}_i^*$ to $\mathcal{T}_i\varphi_i$, and for each $t \in \mathcal{T}_i$ evaluate it starting from $\mathcal{S}_i^* \varphi_i^{-1}$ (this is where we use our unusually precise definition of straight-line programs). This produces some $t^* \in \langle \mathcal{S}_i^* \varphi_i^{-1} \rangle$; check that $t^* t^{-1} \in N_{i+1}$. By (ii) and (iii), we have checked that the factor group $N_i/N_{i+1} \cong C$.

At the end of the induction, we have obtained a correct SGS for the group $N_1 = \langle \mathcal{T}_1 \rangle$ which was output by the composition series algorithm. After that, we verify that $G = N_1$ by sifting the elements of the original generating set $\mathcal{T}$ in N_1.

To justify the nearly linear running time of the entire algorithm, note that m is $O(\log |G|)$ so it is enough to show that the ith step of the induction runs in nearly linear time. We have already seen that the constructions of both $\mathcal{S}_i^*$ and the presentation of $N_i\varphi_i$ are within this time bound. Since both $|\mathcal{T}_i|, |\mathcal{S}_i^*|$ are $O(\log |G|)$, while the length of the presentation is $O(\log^2 |G|)$ and the Schreier-tree data structure of N_{i+1} is shallow, the number of permutation multiplications in (i)–(iii) is bounded from above by a polylogarithmic function of $|G|$.

We note that we have to require that calls to the algorithms in Theorems 1.4, 1.5, and 2.2 fail with probability $< 1/(c \log |G|)$, since during the induction, $O(\log |G|)$ such calls may be made; however, this multiplies the running time only by a $\log \log |G|$ factor. □

Proof of Theorem 1.3 The following result is contained in [BGKLP, Sec. 8]. *If each composition factor H_i of the finite group G has a presentation of length $O(\log^C |H_i|)$ for some $C \geq 2$, then G has a presentation of length $O(\log^{C+1} |G|)$.* The proof in [BGKLP] proceeds by the following steps; we need to show that these can be handled in nearly linear time.

(i) Let L be a lifting of the generators of the composition factors to G. Let M be a subset of L of size $O(\log |G|)$ which also generates G.

(ii) Let S be a subset of G such that any element of G can be reached from S by a straight-line program of length $O(\log |G|)$. Write straight-line programs from M to S.

(iii) Write a presentation for G, and simultaneously write straight-line programs from S to $O(\log^C |G|)$ elements of G. Roughly, these elements are those whose membership in N_{i+1} was tested in (i)–(iii) in the proof of Theorem 1.2. Now the presentation in [BGKLP] is obtained in $O(\log^{C+1} |G|)$ deterministic time.

We saw in the proof of Theorem 1.2 that presentations of the composition factors H_i of a black box group satisfying the restriction of Theorem 1.2 can be obtained having length $O(\log^2 |H_i|)$, using a nearly linear time algorithm. Hence, we shall apply the result of [BGKLP] with the value $C = 2$. Moreover, the generating sets S_i^* of the composition factors constructed in the proof of Theorem 1.2 are such that $\bigcup_i S_i^* \varphi_i^{-1}$ has $O(\log |G|)$ elements. In Proposition 2.4 we shall show that any given $g \in G$ can be reached from $\bigcup_i S_i^* \varphi_i^{-1}$ by a straight-line program of length $O(\log |G|)$, and such a straight-line program can be computed in nearly linear time. This means that we can choose $S := L = M$ in (i) and (ii), so that a presentation of G of length $O(\log^3 |G|)$ can indeed be written in nearly linear time, as indicated in (iii) and as required in the theorem.

Proposition 2.4 *Let $G \leq S_n$ be a permutation group, and suppose that the following have already been computed by Las Vegas algorithms, as in the proof of* Theorem 1.2*: a composition series $G = N_1 \rhd N_2 \rhd \cdots \rhd N_m = 1$, homomorphisms $\varphi_i : N_i \to S_n$ with $\ker \varphi_i = N_{i+1}$, and presentations using generating sets $S_i^* \subset N_i \varphi_i$. Then any $g \in G$ can be reached from $\bigcup_i S_i^* \varphi_i^{-1}$ by a straight-line program of length $O(\log |G|)$, and such a straight-line program can be computed in nearly linear time.*

Proof By induction on $i = 1, 2, \ldots, m$, we will construct a straight-line program of length $O(\log(|G|/|N_i|))$ to some $g_i \in G$ such that $g g_i^{-1} \in N_i$. Let $g_1 := 1$. If g_i has already been obtained for some i, then write a straight-line program of length $O(\log |N_i/N_{i+1}|)$ from S_i^* to $(g g_i^{-1})\varphi_i$. In the case when N_i/N_{i+1} is cyclic or sporadic, this can be done by brute force. In the other cases, we use the isomorphism ψ_i between $N_i \varphi_i$ and a black box group, as in the proof of Theorem 1.2, and the fact that $N_i \varphi_i \psi_i$ is black box m-recognizable. Evaluate this straight-line program starting from $S_i^* \varphi_i^{-1}$, producing an element $h_i \in N_i$. Here, $g g_i^{-1} h_i^{-1} \in N_{i+1}$, and we can define $g_{i+1} := h_i g_i$. Finally, we notice that the procedure runs in nearly linear time, since $m(N_i/N_{i+1}) \leq n$. □

References

[Ba] L. Babai, Local expansion of vertex-transitive graphs and random generation in finite groups, pp. 164–174 in: Proc. ACM Symp. on Theory of Computing 1991.

[BB] L. Babai and R. Beals, A polynomial-time theory of matrix groups and black box groups, in these Proceedings.

[BCFL] S. Bratus, G. Cooperman, L. Finkelstein and S. Linton (in preparation).

[BCFS] L. Babai, G. Cooperman, L. Finkelstein and Á Seress, Nearly linear time algorithms for permutation groups with a small base, pp. 200–209 in: Proc. Int. Symp. Symbolic and Algebraic Computation, ACM 1991.

[BGKLP] L. Babai, A. J. Goodman, W. M. Kantor, E. M. Luks and P. P. Pálfy, Short presentations for finite groups, J. Algebra 194 (1997), 79–112.

[BLNPS] R. Beals, C. R. Leedham-Green, A. C. Niemeyer, C. E. Praeger and Á. Seress, A mélange of black box algorithms for recognising finite symmetric and alternating groups (in preparation).

[BLS1] L. Babai, E. M. Luks and Á. Seress, Permutation groups in NC, pp. 409–420 in: Proc. ACM Symp. on Theory of Computing 1987.

[BLS2] L. Babai, E. M. Luks and Á. Seress, Fast management of permutation groups I. SIAM J. Computing 26 (1997).

[BLS3] L. Babai, E. M. Luks and Á. Seress, Fast management of permutation groups II (in preparation).

[BS] R. Beals and Á. Seress, Structure forest and composition factors for small base groups in nearly linear time, pp. 116–125 in: Proc. ACM Symp. on Theory of Computing 1992.

[Ce] F. Celler, Matrixgruppenalgorithmen in GAP. Ph. D. thesis, RWTH Aachen 1997.

[CF] G. Cooperman and L. Finkelstein, A random base change algorithm for permutation groups, J. Symb. Comp. 17 (1994), 513–528.

[CFL] G. Cooperman, L. Finkelstein and S. Linton, Recognizing $GL_n(2)$ in non-standard representation, pp. 85–100 in [FK].

[CLG1] F. Celler and C. R. Leedham-Green, A non-constructive recognition algorithm for the special linear and other classical groups, pp. 61–67 in [FK].

[CLG2] F. Celler and C. R. Leedham-Green, A constructive recognition algorithm for the special linear group (to appear in Proc. ATLAS Conference).

[CLMNO] F. Celler, C. R. Leedham-Green, S. H. Murray, A. C. Niemeyer and E. A. O'Brien, Generating random elements of a finite group. Comm. Alg. 23 (1995) 4931–4948.

[FK] L. Finkelstein and W. M. Kantor, editors, Groups and Computation II, DIMACS Series in Discrete Math. and Theoretical Computer Science, vol. 28, AMS 1997.

[Ka] W. M. Kantor, Sylow's theorem in polynomial time. J. Comp. Syst. Sci. 30 (1985) 359–394.

[KS] W. M. Kantor and Á. Seress, Black box classical groups (submitted).

[LS] E. M. Luks and Á. Seress, Computing the Fitting subgroup and solvable radical of small-base permutation groups in nearly linear time, pp. 169–181 in [FK].

[Mo] P. Morje, A nearly linear algorithm for Sylow subgroups of permutation groups. Ph.D. thesis, The Ohio State University 1995.

[NiP1] A. C. Niemeyer and C. E. Praeger, Implementing a recognition algorithm for classical groups, pp. 273–296 in [FK].

[NiP2] A. C. Niemeyer and C. E. Praeger, A recognition algorithm for classical groups over finite fields (to appear in Proc. London Math. Soc.).

[NP] P. M. Neumann and C. E. Praeger, A recognition algorithm for special linear groups. Proc. London Math. Soc. (3) 65 (1992), 555–603.

[Pr] C. E. Praeger, Primitive prime divisor elements in finite classical groups, in these Proceedings.

[Ra] F. Rákóczi, Fast recognition of nilpotency of permutation groups, pp. 265–269 in: Proc. Int. Symp. Symbolic and Algebraic Computation, ACM 1995.

[Sch+] M. Schönert et. al., GAP: Groups, Algorithms, and Programming, Lehrstuhl D für Mathematik, RWTH Aachen, 1994.

[SchS] M. Schönert and Á. Seress, Finding blocks of imprimitivity in small-base groups in nearly linear time, pp. 144–147 in: Proc. Int. Symp. Symbolic and Algebraic Computation, ACM 1994.

[Ser] Á. Seress, Nearly linear time algorithms for permutation groups: an interplay