Computing with matrix groups*

William M. Kantor and Ákos Seress

1 Introduction

A group is usually input into a computer by specifying the group either using a presentation or using a generating set of permutations or matrices. Here we will emphasize the latter approach, referring to [Si3, Si4, Ser1] for details of the other situations. Thus, the basic computational setting discussed here is as follows: a group is given, specified as $G = \langle X \rangle$ in terms of some generating set X of its elements, where X is an arbitrary subset of either S_n or $\mathrm{GL}(d, q)$ (a familiar example is the group of Rubik's cube). The goal is then to find properties of G efficiently, such as $|G|$, the derived series, a composition series, Sylow subgroups, and so on.

When G is a group of permutations there is a very well-developed body of literature and algorithms for studying its properties (see Section 2). The matrix group situation is much more difficult, and is the focus of the remaining sections of this brief survey. Sections 4 and 5 discuss the case of simple groups, and section 6 uses these to deal with general matrix groups. We will generally emphasize the group-theoretic aspects of the subject, rather than ones involving implementation in the computer systems GAP [GAP4] or Magma [BCP]. Thus, the word "efficiently" used above will usually mean for us "in time polynomial in the input length of the problem" rather than "works well in practice".

One can ask for the relevance of such questions to finite group theory. Certainly computers have been involved in the construction of sporadic simple groups, as well as in the study of these and other simple groups. We will make a few comments concerning the expected uses in GAP and Magma of the results presented here. However, our point of view includes a slightly different aspect: the purely mathematical questions raised by computational needs have led to new points of view and new questions concerning familiar groups.

2 Permutation groups

We begin with a brief discussion of the case of permutation groups. Here, X is a set of permutations of $\{1, 2, \ldots, n\}$, and then the word "efficiently" will mean "in time

2000 Mathematics Subject Classification: Primary 20D06; Secondary: 20-04, 20P05, 68W30.
* This research was supported in part by the National Science Foundation.

polynomial in the input length $|X| n \log n$ of the problem" ($|X| n \log n$ is roughly the number of keystrokes needed to input X into a computer). The problem is that a small generating set X can specify a very large group G, so large that it is absurd (both from the theoretical and practical points of view) to imagine listing the elements of G.

The development of efficient computer algorithms for permutation groups was begun by Sims [Si1, Si2]. If $G^{[i]}$ is the pointwise stabilizer of $\{1, \ldots, i-1\}$, then

$$G = G^{[1]} \geq G^{[2]} \geq \cdots \geq G^{[n]} = 1 \tag{2.1}$$

where $|G| = \prod_1^{n-1} |G^{[i]} \colon G^{[i+1]}|$ and $|G^{[i]} \colon G^{[i+1]}|$ is the length of the orbit $\mathcal{O}_i$ of i under $G^{[i]}$ and hence is at most n. Sims developed a data structure to find (generators for) all of these subgroups $G^{[i]}$ and orbits $\mathcal{O}_i$ simultaneously and efficiently. This yielded $|G|$ using only elementary group theory: it did not involve structural properties of groups.

The ideas behind the point stabilizer chain construction can also be used for finding many other properties of G, such as the derived series, solvability, and nilpotence, in polynomial time. The application most important from an algorithmic point of view is a *Membership Test: given $h \in S_n$, decide whether or not $h \in G$; and if it is, obtain h from the generating sets of the $G^{[i]}$.*

The above ideas have been implemented in GAP and MAGMA. A detailed description of point stabilizer constructions, and of many other permutation group algorithms, can be found in [Ser2].

3 Matrix groups

We now turn to the case of a group $G = \langle X \rangle$ in which X is an arbitrary set of invertible matrices over some finite field $\mathbb{F}_q$. The questions remain the same: efficiently find properties of G, such as $|G|$, solvability, a composition series, etc. If $X \subset GL(d,q)$ then the input length is $|X| d^2 \log q$ (since $\log q$ bits are required to write each of the d^2 entries of a matrix)[1]. Once again a small generating set X can specify a very large group G.

These problems seem to be very hard. The fundamental difference from the permutation group setting is that there is no longer, in general, a decreasing sequence of subgroups from G to 1 in which all successive indices are "tiny" (as was the case in (2.1)), even with the very generous definition of "tiny" meaning "polynomial in the input length". However, under reasonable additional conditions, and allowing probabilistic components to the algorithms, this has become an actively studied area. Some of the results use the exact representation of G on $\mathbb{F}_q^d$ implicit in the above description (such as eigenvalues, minimal polynomials and so on), but most of those we survey below avoid trying to deal with the exact representation.

First we need to know that random elements can be found. According to an amazing result of Babai [Ba2], *in polynomial time, with high probability one can find independent, nearly uniformly distributed*[2] *random elements of $G = \langle X \rangle \leq$*

[1]Logs are always to the base 2.

[2]Meaning: for all $g \in G$, $(1-\varepsilon)/|G| \leq \mathrm{Prob}(x = g) \leq (1+\varepsilon)/|G|$ for some fixed $\varepsilon \leq 1/2$.

GL(V). This tour de force involves combinatorial methods but nothing about the structure of G; note that $|G|$ is never known here. The results presented in this survey all involve probabilistic estimates, and it is straightforward to have these estimates take the "nearly" part of these "nearly uniform" elements into account; therefore it is convenient to make believe that we actually have uniformly distributed random elements when discussing later results. In practice, a heuristic algorithm from [CLMNO] is used for finding random group elements, and that method is adequate for algorithms in which correctness of the output is ultimately verified (cf. [Ba3]). The points of view in [Ba2] and [CLMNO] are merged in the recent paper [Pak]. Moreover, a new algorithm for random element generation is described in [Co].

A second important tool in almost all of the results below involves the order of an element $g \in \mathrm{GL}(d,q)$. In general, we do not want to assume that we can find $|g|$; for example, testing that an element has order q^d-1 seems to require at least having the prime factorization of the rather large integer q^d-1 (however, compare Theorem 4.3 below and the remarks following it). Nevertheless, it is possible to determine qualitative properties of $|g|$ without actually computing the order. There are algorithms in [NeP1, NiP, KS2] that can be used to decide whether or not $|g|$ is divisible by some *primitive prime divisor*[3] of p^k-1 for a given prime p and given exponent k.

4 Nonconstructive recognition of simple groups

In the matrix group setting, the problem of *recognizing* simple groups began with the following groundbreaking result:

Theorem 4.1 [NeP1] *There is a randomized polynomial-time algorithm which, when given* $0<\varepsilon<1$ *and* $G=\langle X\rangle \le \mathrm{GL}(V)$, *outputs either* "$G$ definitely contains SL(V)" *or* "G does not contain SL(V), *and the probability that the latter assertion is incorrect, given that G does contain* SL(V), *is less than* ε. [4]

Thus, the algorithm gives an answer guaranteed to be correct if it is "Yes", but there is a small probability that the answer "No" will be incorrect. Randomized algorithms that may return an incorrect answer, where the probability of an incorrect output can be controlled by the user, are called *Monte Carlo algorithms*. A special case of Monte Carlo algorithms is the class of *Las Vegas algorithms*: in this case, an incorrect answer can be recognized, so we can achieve that the output is always correct; however, the algorithm may report failure.

The proof of Theorem 4.1 relies heavily on CFSG[5]: the algorithm searches for certain matrices in G that occur with high probability in SL(V), and then uses nonalgorithmic consequences of CFSG to determine the subgroups of GL(d,q) containing such elements. This theorem was followed by others [NiP, CLG1] that

[3]This means that $|g|$ is divisible by a prime divisor of p^k-1 that does not divide p^i-1 whenever $1\le i<k$. Such prime divisors exist for all but a very limited type of pairs p,k [Zs].

[4]In the future we will avoid ε and merely say that such an algorithm succeeds "with arbitrarily high probability".

[5]The classification of the finite simple groups.

decide, similarly, whether or not a given subgroup $G = \langle X \rangle \leq \mathrm{GL}(d,q)$ contains a classical group defined on V as a normal subgroup.

These are *nonconstructive* recognition algorithms, outputting either "G contains a normal classical group", or "G probably does not contain any classical group of $d \times d$ matrices as a normal subgroup". They do *not* tell us how to "get" any given elements of the classical group from the given generating set (e.g., elementary matrices in the situation of the above theorem).

Of course, there is no reason to expect that a quasisimple subgroup G of $\mathrm{GL}(d,q)$ appears in the most natural representation. Even if we have an irreducible representation of $\mathrm{SL}(d,q)$ on some vector space V, the characteristic and dimension of V may very well be quite different from those of the more familiar d-dimensional representation. In order to handle arbitrary matrix groups, this possibility must be taken into account; and when dealing with an unknown representation some of the more standard tools of linear algebra (minimal and characteristic polynomials, eigenvalues and eigenvectors) do not appear to be sufficiently helpful in identifying composition factors of the group being studied.

In general, it would be especially nice to be able to recover the more natural representations from the given "arbitrary" one; we will return to this in the next section. For now, we note that the name of a simple factor can be determined under suitable additional conditions (Theorems 4.2 and 4.3).

Theorem 4.2 [BKPS] *There is a polynomial-time Monte Carlo algorithm which, when given $G = \langle X \rangle \leq \mathrm{GL}(V)$ such that $G/Z(G)$ is isomorphic to a simple group of Lie type of known characteristic p, finds the* name *of $G/Z(G)$.*

Note that the name gives at least one additional piece of information about G, namely $|G/Z(G)|$. The proof of this theorem in [BKPS] handles all situations except for distinguishing the pairs $\mathrm{PSp}(2m,q), \Omega(2m+1,q)$ with q odd and $m \geq 3$, where entirely different techniques were needed [AB]. Our proof is relatively simple (using information already obtained while proving Theorem 4.3 below). We start with a sample of independent (nearly) uniformly distributed random elements of G. We then find the three largest integers $v_1 > v_2 > v_3$ such that a member of the list has order divisible by a primitive prime divisor of one of the integers $p^v - 1$ for $v = v_1, v_2$ or v_3; our sample is chosen large enough so that, with high probability, these are the three largest v such that $|G|$ is divisible by a primitive prime divisor of $p^v - 1$. In a lot of cases, the triple $\{v_1, v_2, v_3\}$ determines the name of G. In the remaining cases, we investigate the occurrence of element orders divisible by two appropriately chosen primitive prime divisors. While this idea is simple enough, it becomes more awkward and detailed if p is a Fermat or Mersenne prime and no such primitive prime divisor greater than 2 occurs. Nevertheless, the algorithm is not complicated, and has already been implemented in MAGMA by Malle and O'Brien.

While the assumption that p is known is a natural one (cf. Section 6), it would be better to be able to avoid this. A result that preceded the previous theorem attempts to do this:

Theorem 4.3 [KS3] *There is a polynomial-time Monte Carlo algorithm which, when given $G = \langle X \rangle \leq \mathrm{GL}(V)$ such that $G/Z(G)$ is isomorphic to a simple group* *of Lie type of unknown characteristic and such that the order of any given element of G can be computed, finds the* name *of $G/Z(G)$.*

The proof of this theorem rests on a nonalgorithmic property of groups G of Lie type in characteristic p. Define a graph $\Gamma(G)$ whose vertices are the prime powers r^a that occur as orders of elements of G, for all primes $r \neq p$ and integers $a > 0$. Prime powers r^a, s^b are joined if and only if G has an element of order $\mathrm{lcm}(r^a, s^b)$ (thus, every vertex of $\Gamma(G)$ has a loop). We say that two vertices of $\Gamma(G)$ are *equivalent* if they have the same neighbors, and denote by $\Delta(G)$ the quotient graph with respect to this equivalence relation, with vertex set $V(\Delta(G))$. We view $\Delta(G)$ as a simple graph (i.e., without loops and multiple edges) and as a *weighted graph*: the *weight* of $v \in V(\Delta(G))$ is the least common multiple of the prime powers in the equivalence class v. This weighted graph usually determines G:

Theorem 4.4 [KS3] *Let G and G^* be finite simple groups of Lie type such that $\Delta(G) \cong \Delta(G^*)$. Then $G \cong G^*$ with some specific exceptions.*

These exceptions include, of course, the pairs $\mathrm{PSp}(2m, q), \Omega(2m+1, q)$ for q odd and $m \geq 3$; additional exceptions are $\mathrm{PSp}(4, q), \mathrm{PSL}(2, q^2)$; $\mathrm{PSp}(6, q), \mathrm{P}\Omega^+(8, q), \Omega(7, q)$; $\mathrm{PSp}(8, q), \mathrm{P}\Omega^-(8, q)$; and $\mathrm{PSL}(3, 2), G_2(2)'$.

Since p is involved in the definition of $\Delta(G)$, how can the above theorem be used to prove Theorem 4.3? This requires additional properties of G:

(i) [GL] If G has characteristic p and is defined over $\mathbb{F}_q$, then the proportion of elements of order divisible by p is at most $5/q$. (We note that a *lower* bound of $2/5q$ for this proportion was proved in [IKS], also motivated by uses in Computational Group Theory.)

(ii) [KS3, Lü] If $r, s \neq p$ are primes such that G has an element of order divisible by $\mathrm{lcm}(r^a, s^b)$, then the proportion of such elements is large (at least $c/(\text{Lie-rank}(G))^3$, for an absolute constant c).

Now the proof of Theorem 4.3 starts by testing all "small" primes p ("small" means bounded from above by an explicit function of the input length) using [KS1, KM] (cf. Theorem 5.3 and the remarks following it) in order to try to find the characteristic of G. (Note that Theorem 4.2 does not quite apply: it is at least conceivable that that theorem could output an answer even if p is not the characteristic of G; and we do not know the probability that this strange possibility might occur.) If this fails then we find a set of suitably many independent random elements of G, and find their orders. This number is chosen so that, by (i), with high probability none of these orders is divisible by p. This number is also chosen so that, by (ii), for every pair r^a, s^b arising in the definition of $\Gamma(G)$, with high probability one of our elements has order divisible by $\mathrm{lcm}(r^a, s^b)$. Using this we determine $\Delta(G)$, and then the name of G.

According to E. O'Brien, in actual computations with matrix groups G using Magma it is standard to find exact orders of elements of G using extensive tables of prime power factorizations of integers of the form $p^k - 1$ for suitable p and k. Therefore, we expect that there will be a version of the above theorem of more than theoretical importance.

Theorem 4.4 leads to a question already alluded to that might make the theorem even more useful in our computational setting. Consider a group H of Lie type over a field of characteristic $r \neq p$. Define a weighted graph $\Delta^p(H)$ for H using the same description as above but with p in place of the correct characteristic r (so that, for example, r is one of the vertices of $\Delta^p(H)$). Then we conjecture that, if $\Delta^p(H) \cong \Delta(G)$ for a group G of Lie type in characteristic p, then $H \cong G$.

Once again we emphasize that the results in this section do not provide any means of calculating with the given matrix group G using its more familiar representations.

5 Constructive recognition of simple groups

As suggested in the preceding section, there is a need for *constructive* recognition algorithms, which allow us to get from our generating set X to any given element of G.[6] These are of fundamental importance when simple groups are used to handle general groups (see the next section).

In the situation of Theorem 4.1, constructive recognition means the following:

Theorem 5.1 [CLG2] *There is a Las Vegas algorithm which, when given $G = \langle X \rangle$ such that $\mathrm{SL}(V) \leq G \leq \mathrm{GL}(V)$, with arbitrarily high probability outputs a new generating set X^* (in terms of X) such that there is a polynomial-time procedure that gets from X^* to any given $g \in G$.*

However, the algorithm in [CLG2] producing X^* does not quite run in polynomial time: there is a factor q in the timing, where V is a vector space over $\mathbb{F}_q$. The corresponding result has also been proved for all classical groups: given $G = \langle X \rangle \leq \mathrm{GL}(d,q)$ having a normal classical subgroup C defined on V, algorithms in [Ce, Bro1, Bro2] output, with high probability, a new generating set X^* such that there is a polynomial-time procedure that gets from X^* to any given $g \in G$. The version of this theorem in [Bro2] handles all symplectic, orthogonal and unitary groups simultaneously in a more or less uniform manner.

It is not known how to get around the factor of q in the timing indicated above without some other type of assumption. In [CoLG] a lovely idea was introduced to avoid this factor: assume the availability of a way to handle *Discrete Logarithms.* Given $\mathbb{F}_q^* = \langle \rho \rangle$ and $\alpha \in \mathbb{F}_q^*$, the Discrete Log Problem asks for an exponent i such that $\alpha = \rho^i$. There are procedures for accomplishing this that are significantly faster than the $O(q)$ time approach that tests all integers with $0 \leq i < q$. Discrete Logs led to the next

Theorem 5.2 [CoLG] *There is a Las Vegas algorithm which, when given $G = \langle X \rangle$ such that $\mathrm{SL}(V) \leq G \leq \mathrm{GL}(V) = \mathrm{GL}(2,q)$, and also given a way to handle Discrete Logs in $\mathbb{F}_q^*$, with arbitrarily high probability outputs a new generating set X^* such that there is a polynomial-time procedure that gets from X^* to any given $g \in G$. The*

[6]More precisely, such that we can find a *straight-line program* from X to any given $g \in G$: a sequence $g_1, \ldots, g_k = g$ with each term either a member of X, the product of two previous terms or the inverse of a previous term.

time requirement is a polynomial of the input length plus the time of polynomially many calls to the Discrete Log subroutine.

This result has been extended in [LGO] to deal with *arbitrary irreducible* representations of $\mathrm{SL}(2,q)$. This extension is fundamental for Theorems 5.5 and 6.1 below.

We next turn to arbitrary representations of classical groups. While we could assume irreducibility, this does not seem to provide useful information about the general situation.

Theorem 5.3 [KS2] *There is a Las Vegas algorithm which, when given $G = \langle X \rangle \leq \mathrm{GL}(V)$ with $G = G'$ and $G/Z(G)$ isomorphic to some (unknown) classical simple group of given characteristic, with arbitrarily high probability finds the classical group C, and outputs a new generating set X^* (in terms of X) together with an injective map $X^* \to C$ that extends to a* constructive *isomorphism $\Psi\colon G/Z(G) \to C$.*

This means that there is a polynomial-time procedure to get to any given $g \in G$ from X^*, and polynomial-time procedures which take any given $g \in G$ or $c \in C$ and find $(gZ(G))\Psi$ or $c\Psi^{-1}$; moreover, it means that if a set X^* and map $X^* \to C$ are output then they are guaranteed to behave as just indicated.

Versions of this theorem are in [CFL], where it was first shown that this type of result could be proved (in the case $G \cong \mathrm{PSL}(d,2)$), and later in [Bra] when $G/Z(G) \cong \mathrm{PSL}(d,q)$ with $d \geq 4, q > 4$. As in Theorem 5.1, the previous theorem does not quite run in polynomial time: once again there is a factor of at least q in the timing. The case of the exceptional groups of Lie type other than ${}^2F_4(q)$ (also assuming a given characteristic) has been close to completion for a few years [KM]; once again the algorithm has an undesirable factor of q in its timing.

Remark 5.4 We stated Theorem 5.3 in its simplest form. It can be extended to handle matrix groups G that have an almost simple classical factor group G/N of given characteristic, *provided that we can test membership in N.* This extension will play an important role in Section 6. So will the fact that there are similar extensions for the exceptional groups [KM] and for the alternating groups [BLNPS]. These and related results are discussed in [Ka2].

The characteristic assumption in the preceding theorem can be removed using Theorem 4.3. When the characteristic is known, the idea behind the theorem is to try to construct an element in a large conjugacy class, one of whose powers is a (long) root element of G (but usually not a long root element of the underlying group $\mathrm{GL}(V)$!); with reasonably high probability[7], an element of G has this property. These root elements and their random conjugates are then used to generate larger subgroups, eventually leading to a subgroup of rank one less than that of G (if G does not already have rank 1).

Combining the Discrete Log results in Theorem 5.2 and its sequel [LGO] with ideas from the proof of Theorem 5.3 and some new ideas (in [Bro2]) has led to algorithms for many classical groups:

[7]But much less than $1/q$, requiring many more that q selections to make it likely that an element of the desired sort is obtained; this is a principal cause of the timing not being polynomial.

Theorem 5.5 [BK, Bro2] *There is an algorithm which, when given $G = \langle X \rangle \leq \mathrm{GL}(V)$ such that $G/Z(G) \cong C = \mathrm{PSL}(d,q)$, $\mathrm{PSp}(2m,q)$ or $\mathrm{PSU}(d,q)$ and $(q, |V|) \neq 1$, and also given a way to handle Discrete Logarithms in $\mathbb{F}_q^*$, with arbitrarily high probability outputs a constructive isomorphism $\Psi\colon G/Z(G) \to C$. The time requirement is a polynomial of the input length plus the time of polynomially many calls to the Discrete Log subroutine.*

The orthogonal groups present additional difficulties, but should be completed in the near future. Analogous constructive isomorphisms for alternating groups are in [BB1, BLNPS, BP]. There are only a bounded number of sporadic groups, so these do not enter into our asymptotic timing questions.

The algorithms announced in Theorems 5.1–5.5 can also be used as Monte Carlo algorithms to decide whether a given group G is such that $G/Z(G)$ is simple of a type indicated in these theorems. As in the case of nonconstructive recognition, the correctness of a "Yes" answer can be verified, but the verification is much more complicated than in the cases covered by Theorem 4.1 and its extensions. Namely, we have to compute a generating set X^{**} and a short presentation in terms of X^{**}, and prove that $G = \langle X^{**} \rangle$ by expressing the original generators of G in terms of X^{**}. A presentation for a quasisimple group G is called *short* if its length[8] is $O(\log^2 |G|)$. Such short presentations are known for almost all simple groups:

Theorem 5.6 [BGKLP, Suz, HS] *For all simple groups except, perhaps, ${}^2G_2(q)$, there is a presentation of length $O(\log^c |G|)$, where $c = 2$; in fact $c = 1$ for most G.*

The proof in [BGKLP] uses simple tricks to adapt the usual Curtis-Steinberg-Tits presentations for these groups when the rank is at least 2, while the cases ${}^2B_2(q)$ and $\mathrm{PSU}(3,q)$ require different ideas to modify the standard presentations for these groups [Suz, HS]. Short presentations have the following nonalgorithmic consequence needed in the proof of Theorem 6.1:

Theorem 5.7 [BGKLP] *Every finite group G with no composition factor of the form ${}^2G_2(q)$ has a presentation of length $O(\log^3 |G|)$.*

The exponent 3 here is best possible.

Although the primary use of constructive recognition algorithms is in computations with matrix groups, they are useful for computing with permutation groups as well. For example, all modern Sylow subgroup algorithms for permutation groups reduce to the case of simple groups [Ka1, Mo, KLM, CCH]. For any given simple permutation group one first determines an explicit isomorphism with a known simple group, and afterwards studies Sylow subgroups of the concrete simple groups. Deterministic algorithms producing such isomorphisms are in [Ka1, KLM]. In the matrix group setting, finding Sylow subgroups should not be difficult, but conjugating one to another may present some difficulties. Another application of constructive recognition algorithms is the computation of maximal subgroups of permutation groups [EH].

[8]The *length* of a presentation $\langle X \mid R \rangle$ is $|X| + \sum_{r \in R} l_X(r)$.

6 General matrix groups

Given $G = \langle X \rangle \leq \mathrm{GL}(d,q)$, there are two basic approaches to exploring properties of G. One of these is a geometric approach, led by Leedham-Green, and commonly called the "The computational matrix group project". This approach uses Aschbacher's classification [Asch] of subgroups of $\mathrm{GL}(d,q)$. (It was first suggested in [Pr] to use Aschbacher's theorem as a guide in the design of what amounted to nonconstructive algorithms for the study of matrix groups.) There are nine categories in this classification, and the goal is to find at least one category to which G belongs. Eight of these categories describe geometric subgroups of $\mathrm{GL}(d,q)$, which means that G preserves some structure associated with the action of G on the vector space $V = \mathbb{F}_q^d$. Moreover, in seven categories, the kernel N of the action on this structure enables us to consider N and G/N acting in smaller dimension, or over a smaller field, or as a permutation group on a small domain. For example, one category consists of irreducible but imprimitive matrix groups. This means that the dimension d can be written as a product $d = ab$, and there is a decomposition $V = V_1 \oplus \cdots \oplus V_a$ into subspaces of dimension b such that G transitively permutes the set $\{V_1, \ldots, V_a\}$; the normal subgroup N is the kernel of this permutation action, and G/N is a transitive permutation group of degree a.

If we can recognize the action of G on the appropriate structure then handling G can be reduced to recursively handling both N and G/N. This reduction bottoms out when a group is a classical group in its natural action (which is the eighth geometric subgroup category of the Aschbacher classification), or G modulo the scalar matrices is almost simple (the ninth category). These two cases are handled by the constructive recognition algorithms for almost simple groups described in Section 5. Note that, even if we have the images of generators of G under a homomorphism φ defined by the action on some geometric structure where $\mathrm{Im}(\varphi)$ is almost simple, usually constructive recognition of $\mathrm{Im}(\varphi)$ is needed in order to obtain generators for $\mathrm{Ker}(\varphi)$.

As a result of extensive research summarized in [LG], there are practical algorithms for recognizing most categories of the Aschbacher classification.

By contrast, the other approach, initiated by Babai and Beals [BB1], tries to determine the abstract group-theoretic structure of G. Every finite group G has a series of characteristic subgroups $1 \leq O_\infty(G) \leq \mathrm{Soc}^*(G) \leq \mathrm{Pker}(G) \leq G$, where $O_\infty(G)$ is the largest solvable normal subgroup of G; $\mathrm{Soc}^*(G)/O_\infty(G)$ is the socle of the factor group $G/O_\infty(G)$, so that $\mathrm{Soc}^*(G)/O_\infty(G)$ is isomorphic to a direct product $T_1 \times \cdots \times T_k$ of nonabelian simple groups that are permuted by conjugation in G; and $\mathrm{Pker}(G)$ is the kernel of this permutation action. Given $G = \langle X \rangle \leq \mathrm{GL}(d, p^e)$, Babai and Beals [BB2] construct subgroups $H_1, \ldots, H_k$ such that $H_i/O_\infty(H_i) \cong T_i$. Having these H_i at hand, it is possible to construct the permutation group $G/\mathrm{Pker}(G) \leq S_k$, which then can be handled by permutation group methods.

The Babai–Beals algorithm is Monte Carlo, and runs in polynomial time in the input length. Contrary to the geometric approach, it does not use the geometry associated with the matrix group action of G. The fact that $G \leq \mathrm{GL}(d, p^e)$ is only used when appealing to a simple consequence of [LS, FT]: if T_i is of Lie type in characteristic different from p, then T_i has a permutation representation of degree polynomial in d.

Combining the Babai–Beals method with constructive recognition algorithms for simple groups, we obtain the following result.

Theorem 6.1 [KS4] *Given $G = \langle X \rangle \leq GL(d, p^e)$, there is a Las Vegas algorithm that computes the following.*

(i) *The order of G.*

(ii) *A series of subgroups $1 = N_0 \lhd N_1 \lhd \cdots \lhd N_{m-1} \lhd N_m = G$, where N_i/N_{i-1} is a nonabelian simple group or a cyclic group for all i.*

(iii) *A presentation of G.*

(iv) *Given any $g \in GL(d, p^e)$, the decision whether $g \in G$, and if $g \in G$, then a straight-line program from X, reaching g.*

The algorithm uses an oracle to compute discrete logarithms in fields of characteristic p and size up to p^{ed}. In the case when all of those composition factors of Lie type in characteristic p are constructively recognizable with a Discrete Log oracle, the running time is a polynomial in the input length $|X|d^2 e \log p$, plus the time requirement of polynomially many calls to the Discrete Log oracle.

The current list of groups recognizable with a Discrete Log oracle is given in Theorem 5.5.

In part (ii) of Theorem 6.1, we construct a series of subgroups that is "almost" a composition series of G. However, some of the cyclic factor groups may not be simple, since we do not assume that we can factor large integers. Using discrete logs seems to be necessary, since already for 1×1 matrix groups $G \leq \mathrm{GL}(1, q)$, finding $|G|$ amounts to solving a version of the discrete log problem in $\mathbb{F}_q^*$. Also, finding and identifying the composition factors, or at least the nonabelian composition factors, seems to be unavoidable, even if the goal is only to compute the order of the input group.

The special case of Theorem 6.1, when the input group is solvable, was already covered a decade ago by the following remarkable theorem of Luks:

Theorem 6.2 [Lu] *Theorem* 6.1 *holds for solvable matrix groups. In fact, there is a* deterministic *algorithm that computes the required output.*

We sketch the proof of Theorem 6.1. Given $G = \langle X \rangle \leq GL(d, p^e)$, the algorithm announced in Theorem 6.1 starts by appealing to the results of [BB2] to compute a composition series for $G/\mathrm{Pker}(G)$, generators for $\mathrm{Pker}(G)$, and generators for some subgroups $H_i \leq \mathrm{Pker}(G)$, $i = 1, 2, \ldots, k$, such that $H_i/O_\infty(H_i) \cong T_i$ for the simple groups T_i involved in $\mathrm{Soc}^*(G)/O_\infty(G) \cong T_1 \times \cdots \times T_k$. Next, we use an extension of Theorem 4.2 to find polynomially many possibilities for the name of the T_i. Given any $g \in H_i$, we can test whether $g \in O_\infty(H_i)$, by testing the solvability of $\langle g^{H_i} \rangle$. This implies that the primitive prime divisor computations necessary for the algorithm in Theorem 4.2 can be carried out in $\overline{T_i} := H_i/O_\infty(H_i)$. If $\overline{T_i}$ is of Lie type then its characteristic is either p or a prime less than d^2 [LS, FT], so we have only polynomially many possibilities for this name.

After that, we replace each H_i by its normal closure in $\mathrm{Pker}(G)$. This step maintains the property that $H_i/O_\infty(H_i) \cong T_i$, but also makes H_i invariant under the conjugation action of $\mathrm{Pker}(G)$.

We now deal with the subgroups $H_1, \ldots, H_k$ sequentially. The conjugation action of $\mathrm{Pker}(G)$ on H_1 also defines a homomorphism $\varphi_1 \colon \mathrm{Pker}(G) \to \mathrm{Aut}(\overline{T_1})$. Using again our ability to test membership in $O_\infty(H_1)$, if $\overline{T_1}$ is *not* of Lie-type in characteristic p, or if $\overline{T_1}$ is defined in characteristic p but over a field of size $q \le d^2$, then the extension of Theorem 5.3, mentioned in Remark 5.4, can be used to construct the kernel $K_1 := \mathrm{Ker}(\varphi_1)$ of this action.

The only remaining possibility is that $\overline{T_1}$ is of Lie type of characteristic p, and the size q of the field of definition is greater than d^2. In this case, the crucial observation is that $H_1/O_\infty(H_1)$ cannot act nontrivially on any elementary abelian section of $O_\infty(H_1)$ that is not a p-group, since then we would have a cross-characteristic representation of $H_1/O_\infty(H_1)$ of degree not allowed by [LS, FT]. Hence the solvable residual H_1^∞ (the last term of the derived series of H_1) is an extension of a p-group by a simple group isomorphic to $\overline{T_1}$. Therefore, in an appropriate basis, which can be found by the Meat-Axe [HR, IL, NeP2], the matrices for the elements of H_1^∞ have the following form:

$$\begin{pmatrix} I & * & * \\ 0 & A & * \\ 0 & 0 & * \end{pmatrix}$$

The blocks in position $(2,2)$ of these matrices define $\overline{T_1}$ modulo scalars. Hence, concentrating on these blocks, we can perform a constructive recognition with a Discrete Log oracle (see Theorem 5.5). After that, as we outlined for the other possibilities for the isomorphism type of T_1, we obtain generators for K_1.

The same procedure is repeated for the conjugation action of K_1 on H_2, constructing its kernel K_2, and so on. Eventually the kernel K_k is a solvable group, which is handled by Luks's methods (see Theorem 6.2).

As the very last step of the algorithm, we construct a presentation for G. This presentation verifies the correctness of the output.

References

[AB] C. Altseimer and A. V. Borovik, Probabilistic recognition of orthogonal and symplectic groups, pp. 1–20 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[Asch] M. Aschbacher, On the maximal subgroups of the finite classical groups, Invent. Math. 76 (1984) 469–514.

[Ba1] L. Babai, Local expansion of vertex-transitive graphs and random generation in finite groups, pp. 164–174 in: Proc. 23rd ACM Symp. on Theory of Computing 1991.

[Ba2] L. Babai, Randomization in group algorithms: conceptual questions, pp. 1–17 in: Groups and Computation II (eds. L. Finkelstein and W. M. Kantor), DIMACS Series in Discrete Math. and Theoretical Computer Science, vol. 28, AMS 1997.

[BB2] L. Babai and R. Beals, A polynomial-time theory of black-box groups I, pp. 30–64 in: Groups St Andrews 1997 in Bath, I (eds. C. M. Campbell, E. F. Robertson, N. Ruskuc, and G. C. Smith), LMS Lecture Note Series 260, Cambridge U. Press 1999.

[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.

[BKPS] L. Babai, W. M. Kantor, P. P. Pálfy, and Á. Seress, Black-box recognition of finite simple groups of Lie type by statistics of element orders, J. Group Theory (to appear).

[BB1] R. Beals and L. Babai, Las Vegas algorithms for matrix groups, pp. 427–436 in: Proc. 34th IEEE Symp. on Found. of Comp. Science 1993.

[BLNPS] R. Beals, C. Leedham-Green, A. Niemeyer, C. Praeger, and Á. Seress, A black-box group algorithm for recognizing finite symmetric and alternating groups, Trans. Amer. Math. Soc. (to appear).

[BCP] W. Bosma, J. Cannon, and C. Playoust, The Magma algebra system I: The user language, J. Symbolic Comput. 24 (1997), 235–265.

[Bra] S. Bratus, Recognition of finite black-box groups, Ph.D. Thesis, Northeastern University 1999.

[BP] S. Bratus and I. Pak, Fast constructive recognition of a black-box group isomorphic to S_n or A_n using Goldbach's Conjecture. J. Symbolic Comput. 29 (2000), 33–57.

[Bro1] P. A. Brooksbank, A constructive recognition algorithm for the matrix group $\Omega(d, q)$, pp. 79–93 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[Bro2] P. A. Brooksbank, Constructive recognition of the finite simple classical groups. Ph.D. thesis, University of Oregon 2001.

[BK] P. A. Brooksbank and W. M. Kantor, On constructive recognition of a black-box PSL(d, q), pp. 95–111 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[CCH] J. Cannon, B. Cox and D. F. Holt, Computing Sylow subgroups in permutation groups. Computational algebra and number theory (London, 1993). J. Symbolic Comput. 24 (1997), 303–316.

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

[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: Groups and Computation II (eds. L. Finkelstein and W. M. Kantor), DIMACS Series in Discrete Math. and Theoretical Computer Science, vol. 28, AMS 1997.

[CLG2] F. Celler and C. R. Leedham-Green, A constructive recognition algorithm for the special linear group, pp. 11–26 in: The atlas of finite groups: ten years on (eds. R. T. Curtis and R. A. Wilson), LMS Lecture Note Series 249, Cambridge U. Press 1998.

[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. in Alg. 23 (1995) 4931–4948.

[CoLG] M. Conder and C. R. Leedham-Green, Fast recognition of classical groups over large fields, pp. 113–121 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[Co] G. Cooperman, Towards a theoretically sound algorithm for random generation in finite groups, preprint.

[CFL] G. Cooperman, L. Finkelstein and S. Linton, Recognizing $GL_n(2)$ in non-standard representation, pp. 85–100 in: Groups and Computation II (eds. L. Finkelstein and W. M. Kantor), DIMACS Series in Discrete Math. and Theoretical Computer Science, vol. 28, AMS 1997.

[EH] B. Eick and A. Hulpke, Computing the maximal subgroups of a permutation group I, pp. 155–168 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[FT] W. Feit and J. Tits, Projective representations of minimum degree of group extensions, Can. J. Math. 30 (1978), 1092–1102.

[GAP4] The GAP Group, Aachen, St Andrews, GAP – Groups, Algorithms, and Programming, Version 4.3, 2002 (`http://www-gap.dcs.st-and.ac.uk/~gap`).

[GL] R. M. Guralnick and F. Lübeck, On p-singular elements in Chevalley groups in characteristic p, pp. 169–182 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[HR] D. F. Holt and S. Rees, Testing modules for irreducibility, J. Austral. Math. Soc. (Ser. A) 57 (1994), 1–16.

[HS] A. Hulpke and Á. Seress, Short presentations for three-dimensional unitary groups, J. Algebra 245 (2001), 719–729.

[IKS] I. M. Isaacs, W. M. Kantor and N. Spaltenstein, On the probability that a group element is p–singular, J. Algebra 176 (1995), 139–181.

[IL] G. Ivanyos and K. Lux, Treating the exceptional cases of the MeatAxe, Experiment. Math. 9 (2000), 373–381.

[Ka1] W. M. Kantor, Sylow's theorem in polynomial time, J. Comp. Syst. Sci. 30 (1985), 359–394.

[Ka2] W. M. Kantor, Simple groups in computational group theory, pp. 77–86 in: Proc. International Congress of Mathematicans, Berlin 1998, Vol. II.

[KLM] W. M. Kantor, E. M. Luks and P. D. Mark, Parallel algorithms for Sylow subgroups, J. Algorithms 31 (1999), 132–195.

[KM] W. M. Kantor and K. Magaard, Black-box exceptional groups of Lie type (in preparation).

[KS1] W. M. Kantor and Á. Seress, Permutation group algorithms via black box recognition algorithms, pp. 436–446 in: Groups St Andrews 1997 in Bath (Eds. C. Campbell et al.), LMS Lectures Notes Series 261, Cambridge U. Press 1999.

[KS2] W. M. Kantor and Á. Seress, Black box classical groups, Memoirs of the Amer. Math. Soc., 149 (2001), No. 708.

[KS3] W. M. Kantor and Á. Seress, Prime power graphs for groups of Lie type, J. Algebra 247 (2002), 370–434.

[KS4] W. M. Kantor and Á. Seress, Algorithms for finite linear groups (in preparation).

[KL] P. B. Kleidman and M. W. Liebeck, The subgroup structure of the finite classical groups, LMS Lecture Note Series 129, Cambridge U. Press 1990.

[LS] V. Landazuri and G. M. Seitz, On the minimal degrees of projective representations of the finite Chevalley groups, J. Algebra 32 (1974), 418–443.

[LG] C. R. Leedham-Green, The computational matrix group project, pp. 229–247 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[LGO] C. R. Leedham-Green and E. A. O'Brien, Constructive recognition of $\mathrm{SL}(2, q)$ (in preparation).

[Lu] E. M. Luks, Computing in solvable matrix groups, pp. 110–120 in: Proc. 33rd IEEE Symp. on Found. of Comp. Science 1992.

[Lü] F. Lübeck, Finding p'-elements in finite groups of Lie type, pp. 249–255 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

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

[NeP1] P. M. Neumann and C. E. Praeger, A recognition algorithm for special linear groups, Proc. London Math. Soc. 65 (1992), 555–603.

[NeP2] P. M. Neumann and C. E. Praeger, Cyclic matrices and the Meataxe, pp. 291–300 in: Groups and Computation III (eds. W. M. Kantor and Á. Seress), The Ohio State Univ. Math. Res. Inst. Publ. 8, Walter deGruyter, Berlin–New York 2001.

[NiP] A. C. Niemeyer and C. E. Praeger, A recognition algorithm for classical groups over finite fields, Proc. London Math. Soc. (3) 77 (1998), 117–169.

[Pak] I. Pak, The product replacement algorithm is polynomial, Proc. 41st IEEEE Symp. on Found. Comp. Science 2000, 476–485.

[Pr] C. E. Praeger, Computation with matrix groups over finite fields, pp. 189–195 in: Groups and Computation (eds. L. Finkelstein and W. M. Kantor), DIMACS Series in Discrete Math. and Theoretical Computer Science, vol. 11, AMS 1993.

[Ser1] Á. Seress, An introduction to computational group theory, Notices AMS 44 (1997), 671–679.

[Ser2] Á. Seress, Permutation Group Algorithms, Cambridge University Press, 2002.

[Si1] C. C. Sims Computational methods in the study of permutation groups, pp. 169–183 in: Computational problems in abstract algebra (ed. J. Leech), Pergamon 1970.

[Si2] C. C. Sims, Computation with permutation groups, pp. 23–28 in: Proc. Symp. Symb. Alg. Manipulation (ed. S. R. Petrick), ACM 1971.

[Si3] C. C. Sims, Group-theoretic algorithms, a survey, pp. 979–985 in: Proc. ICM, Helsinki 1978.

[Si4] Computing with Finitely Presented Groups, Cambridge University Press, 1994.

[Suz] M. Suzuki, On a class of doubly transitive groups, Ann. Math. 75 (1962) 105–145.

[Zs] K. Zsigmondy, Zur Theorie der Potenzreste, Monatsh. Math. Phys. 3 (1892), 265-284.