Recognising Matrix Groups

Max Neunhöffer

Lehrstuhl D für Mathematik
RWTH Aachen

Tucson 2006

All of this is joint work with Ákos Seress.

Lots of others contributed ideas, results, and code.

The Problem

$\mathbb{F}_q$ field with q elements

$\{M_1, M_2, \ldots, M_k\} \subseteq \mathrm{GL}_d(\mathbb{F}_q)$

$G := \langle M_1, M_2, \ldots, M_k \rangle$ finite

Questions

- What is $|G|$?
- What can be said about the isomorphism type?
- Given $g \in G$, write g as product of the M_i (or in terms of some "nice" generating set of G).
- Do all this "efficiently".

We call this "constructive recognition of G".

Variant: $\{\bar{M}_1, \ldots, \bar{M}_k\} \subseteq \mathrm{PGL}_d(\mathbb{F}_q)$, $G := \langle \bar{M}_1, \ldots, \bar{M}_k \rangle$

Straight line programs

Example:

```
# input:
r:= [ a, b, c ];
# program:
r[4]:= r[1]^2*r[2]*r[1]^-2;
r[5]:= r[4]*r[3]^7;
# return values:
[ r[4], r[5]^5 ]
```

Executed with input (a, b, c) this returns:

$$(a^2ba^{-2}, a^2ba^{-2}c^7a^2ba^{-2}c^7a^2ba^{-2}c^7a^2ba^{-2}c^7a^2ba^{-2}c^7)$$

Straight line programs (SLPs)

- only reference earlier results,
- do not contain loops, branches or subroutines, and
- can express long products memory efficiently.

Efficiency

What does "efficiently" mean?

The maximal number of operations necessary is bounded by a (fixed) polynomial in the "input size".

The input size is measured by

- d: size of matrices,
- k: number of matrices, and
- $\log(q)$: size of a field element.

This is called "in polynomial time".

Also the length of the resulting straight line programs should be decent.

$\Longrightarrow$ we use a "nice" generating set

$\Longrightarrow$ this decision shortened SLPs from 500 000 steps down to 500 in examples

Nasty special case

Is there hope?

q large, $d = k = 1$, $M_1 = [\zeta]$ with ζ a primitive root of $\mathbb{F}_q$

Then our task is the Discrete Logarithm Problem

to which there is currently

NO SOLUTION KNOWN in polynomial time in $\log(q)$

$\Longrightarrow$ We work “modulo” this problem.

History

The Matrix Group Recognition Project:

- 1988, Oberwolfach, Joachim Neubüser:
  How to decide, whether $G = \mathrm{GL}_d(q)$?
- 1992, Peter Neumann, Cheryl Praeger:
  Algorithm to decide whether $\mathrm{SL}_d(q) \leq G$.
- 1999, Charles Leedham-Green:
  "Recognising Matrix Groups"
- 2001, William Kantor, Ákos Seress:
  "Computing with Matrix Groups"
- Eamonn O'Brien: Implementation in Magma
- Lots of other people ...

Our Goals:

- A new implementation in GAP
- Go for completely analysed polynomial-time algorithms
- Improve algorithms

What one can do with matrices

With a matrix group $G = \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_d(q)$ we can

- multiply, invert, compare, power up matrices
- execute straight line programs on matrices
- determine the order of a matrix M, i.e. $\min\{n \in \mathbb{N} \mid M^n = 1\}$
- determine the projective order of a matrix M, i.e. $\min\{n \in \mathbb{N} \mid M^n \in \mathbb{F} \cdot \mathbf{1}_d\}$ (scalar matrices)
- find invariant subspaces $0 < W < \mathbb{F}^{1 \times d}$ with $Wg \subseteq W$ for all $g \in G$ or prove irreducibility: "MEATAXE"
- create (pseudo-) random elements
- act with matrices on vectors or on subspaces $\longrightarrow$ gives homomorphisms to permutation groups

Homomorphisms

Try reduction: For $G = \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_d(q)$ find a homomorphism $\varphi : G \to H$ which is

- explicitly computable
- onto some group $H = \langle \varphi(M_1), \ldots, \varphi(M_k) \rangle$ which is "easier to handle"

Assume we can constructively recognise H.

Set $N := \ker(\varphi)$. Then:

- create a (pseudo-) random element g in G
- map g to H via φ
- express $\varphi(g)$ as an SLP S in $\varphi(M_1), \ldots, \varphi(M_k)$
- execute S on $M_1, \ldots, M_k$, get $g' \in G$ s.t. $\varphi(g) = \varphi(g')$
- $\Longrightarrow g^{-1} \cdot g' \in N$
  $\longrightarrow$ this creates a (pseudo-) random element in N

Composition trees

Produce generators of $N := \ker(\varphi)$ and recognise.

Assume that we have recognised H and N constructively.

What does this help for G?

- $|G| = |H| \cdot |N|$
- G has a subgroup N and a factor group H
- We have recognised G constructively!

Get the recursion going . . .

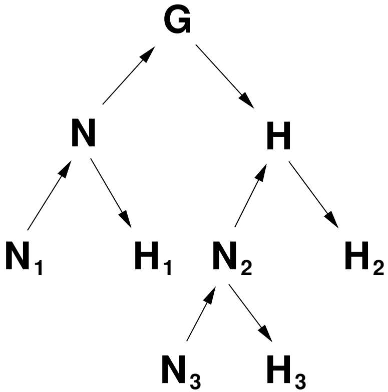
Choose as "nice generators" $M'_1, \ldots, M'_{k'}$ for G:

- preimages under φ of the nice generators of H plus
- the nice generators of N

Given $g \in G$, find an SLP S expressing g in the M'_i:

- map g via φ to $\varphi(g) \in H$
- express $\varphi(g)$ as SLP S' in the nice gens of H
- execute S' on the preimages, get g'
- express $g'^{-1} \cdot g \in N$ as SLP S'' in N
- put together S from S' and S'' plus one multiplication

A Composition Tree



Upward arrows: monomorphisms
Downward arrows: epimorphisms

Low index

Assume:

- G has a maximal subgroup K of low index
- G acts irreducibly
- K leaves a subspace $0 < W < \mathbb{F}_q^{1\times d}$ invariant

Try to find a homomorphism in the following way:

- create random elements, hope they generate K
- find an invariant subspace for these elements
- calculate its orbit under the action of G
- find a homomorphism onto a permutation group H

This works amazingly well!

Unfortunately, it is not yet analysed to be polynomial-time!

Aschbacher's Theorem

Aschbacher classified the maximal subgroups of $\mathrm{GL}_d(q)$.

Theorem (Aschbacher, 1984)

If $G < \mathrm{GL}_d(q)$ then it falls under at least one of:

- **C1** *G leaves invariant a subspace $0 < W < \mathbb{F}_q^{1\times d}$*
- **C2** *G preserves a decomposition $\mathbb{F}_q^{1\times d} \cong V_1 \oplus \cdots \oplus V_j$*
- **C3** *G comes from a bigger field (semilinear)*
- **C4** *G preserves a decomposition $\mathbb{F}_q^{1\times d} \cong V_1 \otimes V_2$*
- **C5** *G is realizable over a subfield*
- **C6** *$G \leq N_{\mathrm{GL}}(r^{1+2k})$ where r^{1+2k} is an extraspecial group*
- **C7** *G is tensor-induced*
- **C8** *G contains a "classical group" like $\mathrm{SL}_d(q)$ or $\mathrm{Sp}_d(q)$*
- **C9** *G is a quasi-simple group*

All classes C1 to C7 are defined "geometrically" and promise some kind of homomorphism or "simplification".

Problem children

The leaves are a problem: Need representation theory.

Classify: Irred. modular representations of finite groups.

This is ongoing research, but there are many results.

We try to

- recognise the "defining characteristic" of the group
- recognise the group for example by looking at distribution of element orders of random elements ("non-constructive recognition")
- use collected data about representations or
- use collected data about subgroups
- directly recognise the group constructively:
  - use base and strong generating sets (matrix Schreier-Sims)
  - use tricks involving involution centralisers

The GAP package recog

GAP already provides:

- the infrastructure for SLPs, matrix handling, etc.
- background algorithms for orbits, MEATAXE, etc.

The recog package provides:

- a completely working framework for composition trees with complete documentation
- a framework to administrate methods to find homomorphisms or leaves
- handling of permutation groups, matrix groups and projective groups in our framework
- switching between different types of groups during recognition

Authors: MN and Ákos Seress

The GAP package recogmethods

The recogmethods package provides:

- asymptotically best algorithms for permutation groups
- methods to find homomorphism for all C1 to C7
- non-constructive recognition of classical groups (C8)
- non-constructive recognition of the defining characteristic of simple groups by the two largest element orders (C9)
- nearly ready non-constructive recognition of simple groups by further element order statistics (C9)
- a start of a database of hints for recognised leaves

Authors: (currently)
Peter Brooksbank, Maska Law, Steve Linton, MN,
Alice Niemeyer, Eamonn O’Brien, Ákos Seress.

Still missing

- analysis of the low index procedure
- some cases in C4 and C7
- constructive recognition after recognising a classical group
  (Charles Leedham-Green and Eamonn O'Brien)
- more hints in the database of hints for recognised leaves
- verification procedures (presentations)
- better methods, maybe "orthogonal" to the Aschbacher classification
- a whole lot of documentation
- higher level algorithms after recognition (Sylow subgroups, maximal subgroups, centralisers, normalisers, etc.)

Help is appreciated

Everybody is welcome to contribute.

We need ideas, code, and analysis.