# Finding normal subgroups

Max Neunhöffer

University of St Andrews

Edinburgh, 17.3.2009

# The problem

**Problem**

*Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and $N$ be a normal subgroup.*
*Produce a non-trivial element of $N$ as a word in the $g_i$ with "high probability".*

- Assume no more knowledge about $G$ or $N$.
- I shall tell you soon why we want to do this.
- We are looking for a randomised algorithm.
- Assume we can generate uniformly distributed random elements in $G$.
- "High probability" means for the moment "higher than $1/[G : N]$".

# Matrix groups ...

Let $\mathbb{F}_q$ be the field with $q$ elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the $M_i$ generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

## What do we want to determine about $G$?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in $G$?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is $G$ simple?
- Comparison with known groups
- (Maximal) subgroups?
- ...

# Constructive recognition

## Problem

Let $\mathbb{F}_q$ be the field with $q$ elements und

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses $M$ as word in the $M_i$.
- The runtime should be bounded from above by a polynomial in $n$, $k$ and $\log q$.
- A Monte Carlo Algorithm is enough. (Verification!)

If this problem is solved, we call

$\langle M_1, \ldots, M_k \rangle$ recognised constructively.

# What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

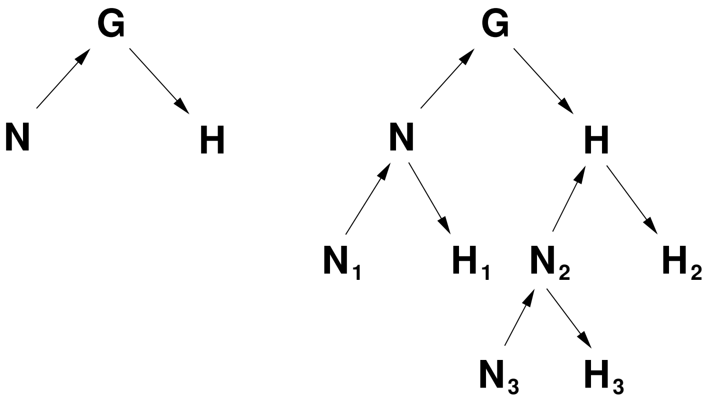
$$\begin{array}{rcccl} \varphi & : & G & \to & H \\ & & M_i & \mapsto & P_i \quad \text{for all } i \end{array}$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- $\varphi$ is surjective: $H = \langle P_1, \ldots, P_k \rangle$
- $H$ is in some sense "smaller"
- or at least "easier to recognise constructively"
- e.g. $H \leq S_m$ or $H \leq \mathrm{GL}_{n'}(\mathbb{F}_{q'})$ with $n' \log q' < n \log q$

# Recursion: composition trees

We get a tree:




Up arrows: inclusions
Down arrows: homomorphisms

Old idea, substantial improvements are still being made

# Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

## Situation

Let $G \le \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that $V$ is irreducible. Assume there is $N$ with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all $W_i$ are invariant under $N$, and $G$ permutes the $W_i$ transitively. Then there is a reduction $\varphi : G \to S_k$.

We can compute the reduction once $N$ is found.

Since we can compute normal closures, our initial problem is exactly, what we need to do.

✓

# Things we can do in matrix groups

We can efficiently:

- store and compare elements
- form products and inverses,
- act on vectors, subspaces and matrices,
- compute element orders
- produce uniformly distributed random elements
- use previously assembled data about groups and representations
- compute normal closures (at least Monte Carlo).

The latter means that for $H < G$, we can compute some elements that generate with high probability the smallest normal subgroup of $G$ containing $H$.

# Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and $G$ non-abelian.

Assume at least one of $x$, $y$ is contained in a non-trivial proper normal subgroup.

We do not know which!

Aim: Produce $1 \neq z \in G$ that is contained in a non-trivial proper normal subgroup.

1. Consider $c := [x, y] := x^{-1}y^{-1}xy$, if $c \neq 1$, we take $z := c$.
2. If $c = 1$, the elements $x$ and $y$ commute. If $x \in Z(G)$, take $z := x$.
3. Compute generators $\{y_i\}$ for $Y := \langle y^G \rangle$.
   - If some $c_i := [x, y_i] \neq 1$, then take $z := c_i$ as in 1.
   - Otherwise $g \in C_G(Y)$ but $g \notin Z(G)$, thus $Y \neq G$, we take $z := y$.

# A first try

## Algorithm 1 (Babai, Beals)

Initialize $1 \neq x := \text{RANDOMELEMENT}(G)$

Repeat $K$ times:

1. $y := \text{RANDOMELEMENT}(G)$
2. $o := \text{ORDER}(y)$
3. $p :=$ some prime divisor of $o$
4. $y' := y^{o/p}$ has order $p$
5. $x := \text{BLINDDESCENT}(x, y')$

Return $x$

✓

# What is the Involution Jumper?

**Input:** $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
**repeat**
  $y := \textsc{RandomElement}(G)$
  $c := x^{-1}y^{-1}xy$ **and** $o := \textsc{Order}(c)$
  **if** $o$ is even **then**
    **return** $c^{o/2}$
  **else**
    $z := y \cdot c^{(o-1)/2}$ **and** $o' := \textsc{Order}(z)$
    **if** $o'$ is even **then**
      **return** $z^{o'/2}$
**until** patience lost
**return** $\textsc{Fail}$

Note: If $xy = yx$ then $c = 1_G$ and $o = 1$ and $z = y$.
But this happens rarely.

# What does the Involution Jumper do?

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.

- If it does not fail, it returns an involution $\tilde{x} \in G$.
- $x\tilde{x} = \tilde{x}x$
- Every involution of $C_G(x)$ occurs with probability $> 0$.
- Using product replacement to produce random elements, this is a practical method for
  - permutation groups,
  - matrix groups and
  - projective groups,

  if nothing goes wrong.
- It needs an involution to start with.

# Jumping between classes

Notation: Let $x^G$ denote the conjugacy class of $x$ in $G$.

**Lemma**

*Let $x, a \in G$ be involutions and $g \in G$. Then*

$$Prob(IJ(x) \in a^G) = Prob(IJ(x^g) \in a^G).$$

or equivalently

**Lemma**

*Let $x \in G$ be an involution. Then the distribution of $IJ(x)^G$ only depends on $x^G$ and not on the choice of $x$ in $x^G$.*

Proof: $f(x, y) :=$

$$\begin{cases} [x,y]^k & \text{if } \mathrm{ORDER}([x,y]) = 2k \\ (y[x,y]^k)^l & \text{if } \mathrm{ORDER}([x,y]) = 2k+1 > 1 \text{ and} \\ & \quad \mathrm{ORDER}([y[x,y]^k]) = 2l \\ y^k & \text{if } xy = yx \text{ and } \mathrm{ORDER}(y) = 2k \end{cases}$$

and we have $f(x^g, y^g) = f(x,y)^g$ whenever $f$ is defined.

# A Markov chain $\mathcal{M}$

The states are the conjugacy classes of involutions in $G$.

The transition is done as follows: At a class $a^G$:

- Pick an arbitrary involution $x \in a^G$.
- Compute $\tilde{x} := IJ(x)$ until $\tilde{x} \neq \text{FAIL}$.
- Next state is $\tilde{x}^G$.

By the lemma, the distribution of the class $\tilde{x}^G$ does not depend on the choice of $x$.

**Theorem**

*The above Markov chain $\mathcal{M}$ is irreducible and aperiodic and thus has a stationary distribution in which every state has non-zero probability.*

# Back to the original question

## Problem

*Let $1 < N \triangleleft G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.*
*Produce a non-trivial element of N as a word in the $g_i$ with "high probability".*

- If we find an involution in $G$ to start with
- and $N$ contains at least one involution class,

the IJ will eventually jump onto an involution class in $N$.

✓

# A better try

## Algorithm 2

Initialize $1 \neq x := \textsc{RandomElement}(G)$ and
$z := \textsc{RandomInvolution}(G)$

Repeat $K$ times:

1. $y := \textsc{RandomElement}(G)$
2. $o := \textsc{Order}(y)$
3. For a few prime divisors $p$ of $o$ do:
   - $y' := y^{o/p}$ has order $p$
   - $x := \textsc{BlindDescent}(x, y')$
4. $z := \textsc{InvolutionJumper}(G, z)$
5. $x := \textsc{BlindDescent}(x, z)$

Return $x$

# Examples

In practice, the IJ works extremely well in many cases:

| $G$ | $N$ | # hops* |
|---|---|---|
| $S_5 \wr S_{10}$ | $S_5^{\times 10}$ | 1.91 |
| $\mathrm{GL}(3,3) \wr S_6 < \mathrm{GL}(18,3)$ | $\mathrm{GL}(3,3)^{\times 6}$ | 1.17 |
| $\mathrm{Sp}(6,3) \otimes 2.\mathrm{O}(7,3) < \mathrm{GL}(48,3)$ | $\mathrm{Sp}(6,3) \otimes 1$ | 1.83 |

* average number of IJ hops needed to reach $N$.

Running Algorithm 2 (with $K = 5$) also works nicely:

| $G$ | $N$ | succ. |
|---|---|---|
| $S_5 \wr S_{10}$ | $S_5^{\times 10}$ | 100% |
| $\mathrm{GL}(3,3) \wr S_6 < \mathrm{GL}(18,3)$ | $\mathrm{GL}(3,3)^{\times 6}$ | 100% |
| $\mathrm{Sp}(6,3) \otimes 2.\mathrm{O}(7,3) < \mathrm{GL}(48,3)$ | $\mathrm{Sp}(6,3) \otimes 1$ | 100% |

(here we have done 100 runs)

# Reductions for imprimitive matrix groups

## Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that $V$ is irreducible. Assume there is $N$ with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all $W_i$ are invariant under $N$, and $G$ permutes the $W_i$ transitively. Then there is a reduction $\varphi : G \to S_k$.

We use Algorithm 2, for the result $x$, do:

- compute the normal closure $M := \langle x^G \rangle$,
- use the MeatAxe to check whether $V|_M$ is reducible,
- if $x \in N$, we find a reduction.

# Possible problems

The InvolutionJumper is in trouble, if at least one of the following happens:

- we do not easily find an involution in $G$ (like for example in $\mathrm{SL}(2, 2^n)$ for big $n$),
- the involution classes of $N$ have a small probability in the limit distribution (when does this happen?),
- the Markov chain does not converge quick enough to its limiting distribution (how quick does it converge?),
- the Involution Jumper returns FAIL too often (when does this happen?),
- $N$ has odd order.

Fortunately: Centralisers of involutions seem to contain enough involutions.