Finding normal subgroups of even order

Max Neunhöffer

University of St Andrews

Bath, 7.8.2009

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.

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

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

- Assume no more knowledge about G or N.

The problem

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

- Assume no more knowledge about G or N.
- We are looking for a randomised algorithm.

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

- Assume no more knowledge about G or N.
- We are looking for a randomised algorithm.
- Assume we can generate uniformly distributed random elements in G.

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.
- We are looking for a randomised algorithm.
- Assume we can generate uniformly distributed random elements in G.

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.
- 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 $1/[G : N]$".

The problem

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".

- Assume no more knowledge about G or N.
- 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 $1/[G : N]$".
- Assume that we can compute in the group and can compute element orders.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Then, for $C := C_G(x)$*, we have:*

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Then, for $C := C_G(x)$, *we have:*

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even. The orbits of $\langle x \rangle$ on N have lengths 1 and 2, so there must be an even number of orbits of length 1. ∎

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even. The orbits of $\langle x \rangle$ on N have lengths 1 and 2, so there must be an even number of orbits of length 1. ∎

In particular, $C \cap N$ contains an involution.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even. The orbits of $\langle x \rangle$ on N have lengths 1 and 2, so there must be an even number of orbits of length 1. ∎

In particular, $C \cap N$ contains an involution.

That is, we can replace (N, G) with $(C \cap N, C)$ and use the statement again, provided we find another non-central involution.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.

Finding $N \triangleleft G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

Finding $N \triangleleft G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

We find involutions by powering up random elements.

Involution centralisers

How can we compute the centraliser of an involution?

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
$y := \textsc{RandomElement}(G)$
$c := x^{-1}y^{-1}xy$ **and** $o := \textsc{Order}(c)$

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
  $y := \text{RandomElement}(G)$
  $c := x^{-1}y^{-1}xy$ **and** $o := \text{Order}(c)$
  if o is even **then**
    append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
  else
    append $z := y \cdot c^{(o-1)/2}$ to *gens*

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
  $y := \text{RandomElement}(G)$
  $c := x^{-1}y^{-1}xy$ **and** $o := \text{Order}(c)$
  if o is even **then**
    append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
  else
    append $z := y \cdot c^{(o-1)/2}$ to *gens*
until o was odd often enough or gens long enough
return *gens*

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
    $y := \text{RandomElement}(G)$
    $c := x^{-1}y^{-1}xy$ **and** $o := \text{Order}(c)$
    if o is even **then**
        append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
    else
        append $z := y \cdot c^{(o-1)/2}$ to *gens*
until o was odd often enough or gens long enough
return *gens*

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

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
    $y := \text{RandomElement}(G)$
    $c := x^{-1}y^{-1}xy$ **and** $o := \text{Order}(c)$
    if o is even **then**
        append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
    else
        append $z := y \cdot c^{(o-1)/2}$ to *gens*
until o was odd often enough or gens long enough
return *gens*

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

And: If o is odd, then z is uniformly distributed in $C_G(x)$.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

Finding $N \lhd G$

We want to find an N with $1 < N \unlhd G$ and $2 \mid |N|$, or conclude that there is none.

We can proceed as follows: Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

How do we test if we have a proper normal subgroup?

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.
initialise $o := \text{Order}(g)$
for $i := 1$ to 20 **do**
    $y := \text{RandomElement}(N)$

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.
initialise $o := \textsc{Order}(g)$
for $i := 1$ to 20 **do**
    $y := \textsc{RandomElement}(N)$
    $o := \textsc{Gcd}(o, \textsc{Order}(yg))$

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.
initialise $o := \text{Order}(g)$
for $i := 1$ to 20 **do**
  $y := \text{RandomElement}(N)$
  $o := \text{Gcd}(o, \text{Order}(yg))$
  if $o = 1$ **then**
    return 1

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.
initialise $o := \text{Order}(g)$
for $i := 1$ to 20 **do**
  $y := \text{RandomElement}(N)$
  $o := \text{Gcd}(o, \text{Order}(yg))$
  if $o = 1$ **then**
    return 1
return o

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.
initialise $o := \textsc{Order}(g)$
for $i := 1$ to 20 **do**
  $y := \textsc{RandomElement}(N)$
  $o := \textsc{Gcd}(o, \textsc{Order}(yg))$
  if $o = 1$ **then**
    return 1
return o

This is a one-sided Monte Carlo algorithm.

Testing for a proper normal subgroup

The following method by Charles Leedham-Green estimates the order of $gN \in G/N$:

Algorithm: EstimateOrder

Input: $g \in G$ and a $N = \langle n_1, \ldots, n_m \rangle \trianglelefteq G$.
initialise $o := \textsc{Order}(g)$
for $i := 1$ to 20 **do**
  $y := \textsc{RandomElement}(N)$
  $o := \textsc{Gcd}(o, \textsc{Order}(yg))$
  if $o = 1$ **then**
    return 1
return o

This is a one-sided Monte Carlo algorithm.

We estimate all orders $g_i N \in G/N$ to decide $G = N$.

The method in action

We look at the following examples:

- $S_{30} \wr S_7 < S_{210}$ (imprimitive action)
- 3rd maximal subgroup of M_{24} on 24 points: $2^4 : A_8$
- 5th maximal subgroup of M_{24} on 24 points: $2^6 : 3.S_6$
- Double cover $2.Suz$ of the sporadic Suzuki group
- $Sp(6,2) \wr S_6 < GL(36,2)$ (imprimitive)
- $SL(6,3) \circ M12 < GL(10,3)$ in $GL(60,3)$ (tensor decomposable)

What can go wrong?

What can go wrong?

Actually, lots of things!

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.
- We could get a normal closure wrong.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.
- We could get a normal closure wrong.
- We could get an order estimate wrong.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.
- We could get a normal closure wrong.
- We could get an order estimate wrong.
- G might not have an even order normal subgroup.

✓