THE UNIVERSITY OF WESTERN AUSTRALIA

Achieving International Excellence

Estimation Problems and Randomised Group Algorithms

Cheryl E Praeger

Galway, April, 2011

The five linked lectures

- Introduction: estimation/randomisation [Cheryl]
- Estimation and algorithms in Permutation groups [Alice]
- Estimation techniques in Lie type groups [Cheryl/Alice]
- Involutions and regular semisimple elements - serendipity [Cheryl]
- Classical group generation by balanced involutions [Akos]

Randomisation - Why?

THE UNIVERSITY OF WESTERN AUSTRALIA
Achieving International Excellence

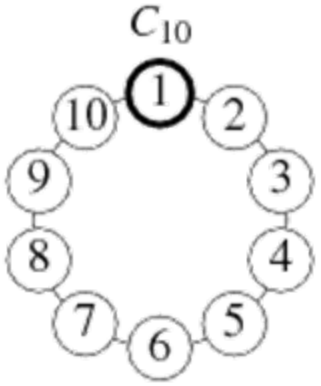
Some potted history

- Charles Sims' permutation group algorithms

Base of permutation group

- Example $G=D_{2n} = < a=(12... n), b=(2n)(3,n-1)...>$
- Base B=(1,2) Only identity of G fixes all points of B
- Represent each of the 2n elements g of G by image of B under g
- Example a maps B to (2,3), b maps B to (1,n)
- Small bases give compact [space/time saving] representations of group elements

Sims' ingenious methods computes using base images



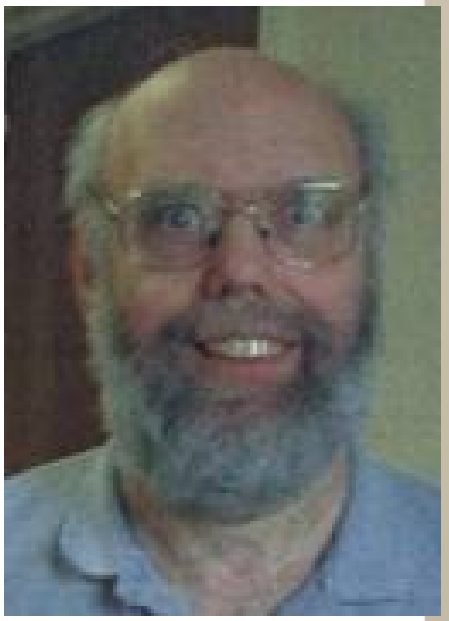
Still – Why randomisation?

Usefulness [around 1970]

- Sims proved existence of Lyons sporadic simple group by constructing it on a computer which could not even store and multiply the two generators in their smallest degree permutation representation! He needed to use base images

So what's the problem?

- Sims general purpose perm group algorithms great
- Except when minimum base size too large
- The Giants: Sn and An
- Base for Sn – (1,2, ..., n-1)
- Base for An – (1,2, ..., n-2)



John Cannon and CAYLEY 1970s

- Given G = < X > permutation group with gen'g set X
  - If we know G is An or Sn have special methods
  - If G is not An or Sn and G is primitive then G has a much smaller base and Sims' methods worked brilliantly [for computations then]
- So how to identify the giants An and Sn?
  - Use theory from 1870s
  - Many elements ONLY exist in giants
  - So many that we should find them with high probabiltiy by random selection in a giant

Jordan's Theorem circa 1870

- Given transitive permutation group G < S_n, a prime p such that n/2<p< n-2
- If some element of G contains a p-cycle then G is An or Sn

How useful is this?

How common are Jordan's 'good' elements?

Define: $g \in S_n$ is **'good'** if g contains a p-cycle, for some prime p, $n/2 < p \leq n-3$

Example: $g = (12345)(67) \in S_9$ is 'good': $n = 9, p = 5$

For fixed p: number of elements in S_n containing a p-cycle is

$$\binom{n}{p}(p-1)!(n-p)! = \frac{n!}{p} \quad \text{(and} \quad \frac{n!}{2p} \quad \text{in } A_n\text{)}$$

Proportion of 'good' elements in A_n or S_n: $\sum_{n/2<p\leq n-3} \frac{1}{p} \geq \frac{c}{\log n}$ for some constant c

So roughly c from every log n elements is “good”

How do we turn this into a justifiable algorithm?

- Randomly select some multiple of (log n)/c elements from a transitive group G on n points (which is Sn or An, but this knowledge is a secret) and
- expect to find a “good” Jordan element, thereby revealing the secret that G really is a giant Sn or An.

- A Monte Carlo algorithm!

Monte Carlo algorithms

- named after Monte Carlo Casino in Monaco
- where physicist Stanislaw Ulam's uncle used to borrow money to gamble

want the algorithm to complete quickly, allow a small (controlled) probability of error.

Monte Carlo algorithm

- For true Monte Carlo algorithm: estimate bound on likely error size based on number N of random selections;
- Alternatively: for a given acceptable error e estimate likely number of random selections N = N(e) needed to achieve this
- Famous uses:
  - Enrico Fermi (1930) the properties of the neutron
  - Los Alamos (1950s) for early work on hydrogen bomb

Monte Carlo algorithm to recognise S_n, A_n

Input: Transitive $G = \langle x_1, \ldots, x_k \rangle \leq S_n$ and real number ε ($0 < \varepsilon < 1$, error probability bound)

Output: True (hopefully if G is S_n or A_n) or False

Algorithm: Select up to $N = \lceil (\log \varepsilon^{-1})(\log n)/c \rceil$ random elements g from G and test if g is 'good'.

If a 'good' element is found then return True
If no 'good' elements are found then return False

What does this algorithm actually do?: (At least it completes!)

1. If the algorithm returns True then $G = A_n$ or S_n (**guaranteed by Jordan's Theorem**)

2. If the algorithm returns False then this may be incorrect, but only if G does equal A_n or S_n, and we failed to find a 'good' element.

3. **Prob**(do not find good element, **given that** $G = A_n$ **or** S_n)

$$\leq \left(1 - \frac{c}{\log n}\right)^N < \varepsilon$$

So this is a **Monte Carlo algorithm with error probability less than** ε.

Further Comments on Context

1: assume available approximately independent **random elements** from G (Both theoretical and practical algorithms exist for this.)

2: Monte Carlo algorithms: error probability must be controlled

3: variety of mathematics required for both design and proof

Algebra	to prove correctness of output
Probability estimates	to control error

- This is `essentially' algorithm used in GAP and MAGMA for testing if G is a permutation group giant. Developed by John Cannon.
- Cannon's algorithm relies on generalisations of Jordan's Theorem due to Jordan, Manning, CEP and others. Use a larger family of `good‘ elements.

Notice the role of estimation:

lower bound for proportion of “good” elements leads to upper bound on error probability

How good an estimate?

If estimate is far from true value does it matter?

- Yes and No !
- No: because if there are more good elements than we estimate then we just find them more quickly and algorithm confirms “G is a giant” more quickly
- Yes: because if G is not a giant then we force the algorithm to do needless work in testing too large a number of random elements [it will never find a good one] and so the algorithm runs too slowly!

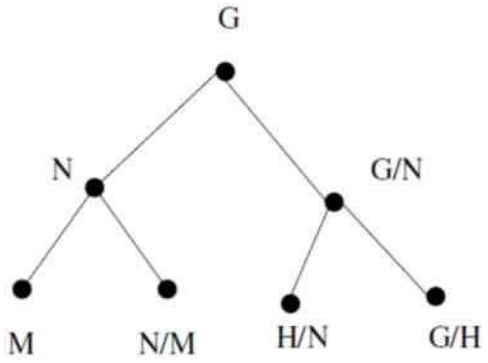
So the upshot is: it really does matter. We should try to make estimates as good as possible, especially when they are for an algorithmic application.

Focus on simple groups

Few general statements on group computation

'Tree View' underpins new generation of group algorithms:

Focus on finite simple groups:



Some names: O'Brien, Leedham-Green, Seress, Neunhoeffer, . . .

Example from classical groups

Class$(n, q) = \mathrm{GL}(n, q), \mathrm{Sp}(n, q)$ etc acting on $V = V(n, q)$

Primitive prime divisor (ppd) of $q^e - 1$ a prime r dividing $q^e - 1$ such that $\nexists\, i < e$ with r dividing $q^i - 1$

Ppds interesting because superficially

$$|\mathrm{Class}(n, q)| = q^{\text{some power}} \prod_{\text{various } i} (q^i - 1)$$

ppd-$(n, q; e)$ element $g \in \mathrm{Class}(n, q)$ is an element with order divisible by a ppd of $q^e - 1$;

"good ppd element": $e > n/2$ plus minor additional conditions

Importance of ppds in classical groups: ppd Classification Theorem 1998

For an irreducible subgroup G of Class(n, q), if G contains “two good ppd elements” then essentially G = Class(n, q) with SMALLLIST of exceptions

[with Alice Niemeyer]

Deep result – proof relies on simple group classification

Classical recognition algorithm 1998 [NieP]

Input: $G = \langle X_1, \ldots, X_k \rangle \leq \mathrm{Class}(n, q)$

Output: True (and then sure that $G = \mathrm{Class}(n, q)$), or False.

Classical Recognition Algorithm: Niemeyer, CEP, 1998

1. Test MANY random elements of G;
2. If "two good ppd elements" not found return False;
3. If found and test for membership in SMALLLIST positive, return False;
4. Else report True

But how many is MANY?

Is it a Monte Carlo algorithm?

- If it returns True then G really is $\mathrm{Class}(n, q)$ (by theorem)
- If it returns False this may be incorrect
  (namely if $G = \mathrm{Class}(n, q)$ and we fail to find good ppds).

If we knew the proportion of "good ppd pairs" in $\mathrm{Class}(n, q)$ then we could estimate how many random elements to test – Monte Carlo Algorithm

Basic problem: Estimate the proportion of good ppd elements in $\mathrm{Class}(n, q)$.

First the answer:

For $G = \mathrm{Class}(n, q)$ and $e > n/2$ let $\mathrm{PPD}(G, e)$ be the proportion of ppd-$(n, q; e)$ elements in G

Adding over all such e let $\mathrm{PPD}(G)$ be proportion of ppd elements in G

ppd Estimation Theorem: Niemeyer, CEP, 1998

Let $e > \frac{n}{2}$ such that $q^e - 1$ divides $|G|$. Then

(a) $\frac{1}{e+1} \leq \mathrm{PPD}(G, e) \leq \frac{1}{e}$.

(b) $\log 2 - \frac{2}{n} \leq \mathrm{PPD}(G) \leq \log 2 + \frac{2}{n}$

[or half this for some types of classical groups]

The Proof uses geometry and group theory

- Case G=GL(n,q) – others similar
- For fixed e first find PPD(G,e) same as for G=GL(e,q)
- So want proportion of elements in GL(e,q) with order multiple of ppd of q^e-1
- Show this is (1/e) x (proportion of such elements in cyclic group of order q^e-1)