Involution centralizers in matrix group algorithms

Robert A. Wilson

QMUL, 14/11/05; Warwick, 17/02/06

Introduction In this talk all groups are finite (by definition), and all simple groups are non-abelian. Let us first define our terms: an *involution* in a group G is an element t of order 2, i.e. $t^2 = 1$ and $t \neq 1$. Its *centralizer* $C_G(t)$ is the subgroup of elements which commute with it, so $C_G(t) = \{g \in G \mid tg = gt\}$. It has long been accepted in abstract group theory that the way to study simple groups is via their involution centralizers. But this orthodoxy has been slow to filter through into computational group theory. Part of the purpose of this talk is to advertise the usefulness of involution centralizer methods in this field.

Finding the centralizer of an involution Back in the 1970s and 1980s, Cambridge-style computational group theory was using involution centralizers, using the fact that two involutions t and u always generate a dihedral group, and that if tu has even order, $2k$ say, then $(tu)^k$ is the central involution in this dihedral group. In particular, $(tu)^k$ commutes with t, and we hope that by picking enough involutions u we will find enough elements to generate $C_G(t)$.

Bray's method This method was adequate for purposes at the time, but strangely enough it was twenty years before we realised we were throwing away the most useful part of the information: the case when $\langle t, u\rangle$ has *odd* order, say $2k+1$. For in this case, $(tu)^k$ conjugates u to t (check it if you don't believe me: $(tu)^{-k}u(tu)^k = (ut)^k u(tu)^k = (ut)^{2k+1}t = t$), and if we take u to be a conjugate of t, say $u = t^x$, then $x(tu)^k$ conjugates t to u and back to t again. In other words $x[t,x]^k \in C_G(t)$. We credit John Bray with bringing this to our attention. (It is also worth noting that in the even order case, we not only have $[t,x]^k \in C_G(t)$, but also $[t,x^{-1}]^k \in C_G(t)$, so we get two elements for the price of one.)

But it is even better than this. Fix x with $[t,x]$ of order $2k+1$, and let y run through the elements of $C_G(t)$. Then $[t,yx] = [t,x]$ so $(yx)[t,yx]^k = y(x[t,x]^k)$ runs through the elements of $C_G(t)$ also. In other words, if x is a *random* element of G with the property that $[t,x]$ has odd order, then $x[t,x]^k$ is a *random* element of $C_G(t)$. This observation, due to Richard Parker, means we can actually prove useful results about how long it will take to get enough elements to generate the involution centralizer.

Finding an involution So we can find the centralizer of an involution effectively. Or can we? First we have to find the involution! The only general method for finding an involution is to find an element of even order, say x of order $2k$, and compute the involution x^k. Unfortunately, some groups don't have very many involutions (for example, groups of odd order!). But seriously, the simple group $SL_2(2^{1009})$ is a group which it is perfectly possible to compute with, but the proportion of its elements which have even order is roughly 2^{-1009}, which is so small that the lifetime of the universe is far too short a time in which to have the slightest chance of ever seeing such an element. (The National Lottery is a dead cert in comparison!)

Ryba's algorithm As an example of what can be done with the techniques developed by the Cambridge 'school', here is a method known as Ryba's algorithm, although it is more a 'method' than an 'algorithm' in the technical sense. It 'solves' the problem of 'constructive membership testing'. Suppose we have a group H lying in some ambient group G, and we wish to test whether $H = G$, or more particularly, for any element $g \in G$, we want to test if $g \in H$. Suppose H is given by a set of generators $\{h_1, \ldots, h_r\}$, and we also want to write g as a word in $h_1, \ldots, h_r$.

- Find $h \in H$ such that gh has even order, $2k$, say, and let $x = (gh)^k$ be the involution it powers to.
- Find an involution $z \in H$ such that xz has even order, $2m$, say, and let $y = (xz)^m$ be the involution it powers to.
- Construct $C_H(z)$ and test if $y \in C_H(z)$ (equivalently, $y \in H$). If so, return a word for y in the generators of H. Otherwise, return 'false'.
- Construct $C_H(y)$ and test if $x \in C_H(y)$ (equivalently, $x \in H$). If so, return a word for x in the generators of H. Otherwise, return 'false'.
- Construct $C_H(x)$ and test if $gh \in C_H(x)$ (equivalently, $g \in H$). If so, return a word for g in the generators of H. Otherwise, return 'false'.

In principle this is a recursive algorithm, but there are technical difficulties with implementing the recursion, for example dealing with the base case. Moreover, in the worst case the algorithm is exponential-time.

Implementations exist for sporadic groups by Holmes and Linton, and for classical groups in odd characteristic by O'Brien and Wilson. The latter includes improvements which choose the involutions x, y and z in such a way that their eigenspaces on the natural module have dimensions close to half the dimension of the whole space, in order to bound the depth of the recursion. It calls a 'composition tree' algorithm to deal with the recursive call, which is certainly not the most efficient way to do it, but it does work.

Naming simple groups Suppose we have a group which is known to be simple. How do we tell which simple group it is? One well-known method is to calculate the orders of a random sample of group elements, and use statistical methods. But this does not distinguish between $PSp_{2n}(q)$ and $P\Omega_{2n+1}(q)$, which have the same order, and the same statistics of element orders. Altseimer and Borovik propose a method which involves finding an involution in a particular conjugacy class (or in one of two specified conjugacy classes), finding its centralizer, and identifying the non-abelian composition factors in this centralizer. Since these composition factors are different in the two cases (and it does not just recurse to a smaller problem of the same kind!), this distinguishes them.

A challenge problem The problem of recognising a simple group, if it is not known in advance to be simple, is much harder. Babai and Shalev reduce (in polynomial time) to the case of distinguishing between a simple group S and an extension VS by an irreducible S-module V. For example, the 'challenge problem' of Babai and Shalev asks for a method of distinguishing between the simple group $\Omega_4^-(p) \cong PSL_2(p^2)$ and $p^4\Omega_4^-(p)$, where p is a very large prime. (Actually, they ask a more general question, to distinguish between $PSL_2(p^f)$ and $p^{4f}PSL_2(p^f)$, but this special case is enough to illustrate the method.) The problem here is that almost all elements of $\Omega_4^-(p)$ act fixed-point-freely on the natural module, and therefore we cannot find elements of the normal subgroup by a purely random search.

Chris Parker and I realised immediately that these groups are distinguished by their involution centralizers, of shapes D_{p^2-1} and $p^2D_{p^2-1}$ respectively. Now in a dihedral group, two random commutators commute, whereas in the second group they do not. Does this give rise to a polynomial-time algorithm? Well, in $PSL_2(p^2)$ at least (roughly) a quarter of the elements have even order, so finding an involution is easy. Moreover, the product of two involutions has odd order at least (roughly) half the time, so Bray's method produces (nearly uniformly distributed) random elements of the centralizer. It is not even necessary to generate the whole involution centralizer, as the method only requires four random elements.

Recognising simplicity Clearly this method is capable of generalisation. Using the Babai–Shalev reductions, it should be possible to recognise any simple group of Lie type over a field of odd order, in Monte Carlo polynomial time, among *all* black-box groups of characteristic p. The method is to reduce to an involution centraliser, using the fact that an involution in a simple group S centralizes a non-trivial subspace of any S-module. More importantly, it uses the structures of the involution centralizers in the groups of Lie type to prove that (if G is a simple group of Lie type in odd characteristic, then) G has a non-trivial normal p-subgroup *if and only if* any involution centralizer in G has a non-trivial

normal p-subgroup.

As in the case of Ryba's algorithm, there are practical difficulties with implementing the recursion, since an involution centralizer in a simple group is not simple!

In effect, this method looks for elements of $O_p(G)$, so it should not take much more to produce an algorithm which constructs $O_p(G)$, at least in the sense of providing normal generators. (It may not be practical to provide generators for it as an abstract group, as it may require a very large number of generators.)

Complexity The goal is to produce algorithms which are Monte Carlo polynomial time. Proving that the complexity is as good as this requires detailed knowledge of involutions, and their products, in simple groups. In particular we need estimates of the probability that the product of two random conjugates of an involution has odd order. In practice we need this not just for one or two classes of involutions (as Altseimer and Borovik required), but for a fixed proportion of the conjugacy classes (or preferably all of them!). Chris Parker and I have proved bounds of the form c/r^3, where r is the Lie rank. Of course, in practice we use the even order case of Bray's method as well, and the actual time to run the algorithm is much better than these conservative estimates suggest.